University of Exeter
Department of Computer Science

CONTEXTUAL INFORMATION FOR OBJECT DETECTION

Faisal A. Alamri

July, 2020

Supervised by Dr Nicolas Pugeault & Dr Jacqueline Christmas

Submitted by Faisal A. Alamri, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science , July, 2020.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

(signature) ALAMRI

Abstract

Object detection has improved very rapidly in the last decades, but because they are very essential and considerably needed in various applications, further enhancement is needed. This thesis proposes the use of contextual information captured from digital scenes as a tool to contribute to developing detection performance. Contextual information, such as the co-occurrence of objects and the spatial and relative size among objects, provides deep and complex knowledge and interpretation about scenes. Determining such relationships among objects is seen to provide machine learning models with vital cues that aid detection methods to reach a better performance.

In this thesis, sixteen contextual object-object relationships captured from MSCOCO 2017 training dataset are proposed. Upon the unique and intelligent enlightenment that those sixteen relationships provide, two contextual models, named *Rescoring Model*, and *Relabelling Model*, are proposed. These models explicitly encode contextual information from scenes, resulting to an improvement in the performance of two of the state-of-the-art detectors (*i.e.*, Faster RCNN and YOLO). These models even provide greater improvement when being repeatedly processed, achieving higher AUC, mAP and F1 scores, with an increase of up to 19 percentage points compared with the baseline detectors.

Due to the enhancement those contextual models achieve, another contextual model, named *Transformer-Encoder Detector Module*, is proposed. In contrast to the previous models, this model implicitly encodes contextual statistics and uses attention mechanism to provide a deeper understanding of images contents. It also achieves higher mAP, F1 scores and AUC average score of 13 percentage points compared to Faster RCNN detector.

Perturbed images, where two different approaches of perturbations are applied, are used to examine the impact of the proposed contextual models. Results show that contextual models also gain better performances compared to the baseline detector. This is due to the use of both visual and contextual features, unlike the detector, which depends only on visual features.

I dedicate this thesis to:
My Family: My grandparents, parents, brother and wife
for their love, endless support and encouragement.

Acknowledgements

First and foremost, praises and thanks to *Allah, the Almighty,* for the support, strength, capability and the blessings I have been given to make a dream come true. Being a doctoral student, and submitting a PhD thesis have been the dream since when I was 18, a high school student.

I earned my high school degree with distinction, aiming to travel abroad for higher education. Yes, I was accepted, and came to the UK in 2008. I would like to express my gratitude to *King Abdullah bin Abdulaziz* for funding the Saudi Ministry of Education, which gave me this opportunity, and for the funding and support that I have been receiving until now. In 2013, I earned my Bachelor's degree. A year later, I completed my Masters, as I needed to come closer to my dream.

In 2015, the journey towards my dream started when I was accepted to do a PhD at The University of Exeter. The physical and technical contribution of *'Computer Science Department at The University of Exeter'* is deeply appreciated. Without their acceptance, support and funding, this thesis would not have reached this stage.

My sincerest and heartfelt appreciation goes to my supervisor, *Dr Nicolas Pugeault*, who has not just been my advisor, but more importantly my friend, mentor and instructor. He has been the substance of a genius: he guided, encouraged and motivated me to be professional and to do the impossible. He has been there when needed, unstinting in his support and assistance, which ultimately helped me attain my goal.

My family is my backbone. My Grandparents, *Ghaithan Alamri* and *Sarraa Alamri*, are my inspiration. My warmest love and thanks to them for all the support, motivation and care they have been providing since I was born. My deepest gratitude to my brother, *Sultan Alamri*, who has always been my role model. Thanks to him for his financial and emotional support, which kept me motivated. This work would not have been possible without his input. I wish also to express my boundless and greatest appreciation and thanks to my wife, *Sahar Alamri*, for her love, assistance, encouragement, continuing support, and prayers to complete this project. She brightens my days, makes sacrifices to help and keep me focused in the face of pressure throughout this entire process of doing my PhD.

I wish to thank all the other family members, my parents, brothers and sisters, all my friends and other members of my supervisory team, whose assistance was crucial to the completion of this project. Without their consistent and constant support and help, this project would not have been significantly easier.

Finally, my appreciation to the examiners, *Dr Frank Guerin* and *Dr Anjan Dutta*, for undertaking to examine this research work, and for any feedback that they may provide to enhance it, and for making my dream come true.

Contents

List of Tables

List of Figures

Nomenclature and Abbreviations

Acronyms and abbreviations

AP	Average Precision.
AUC	Area Under the Curve.
CNN	Convolutional Neural Network.
DNN	Deep Neural Network.
DPM	Deformable Part-Based Model.
Faster RCNN	Faster Region-Based Convolutional Neural Networks.
FFF	Fast Feature Fool.
FN	False Negative.
FP	False Positive.
FPR	False Positive Rate.
IoU	Intersection Over Union.
IRM	Iterated Relabelling Model.
LSTM	Long Short-Term Memory Network.
mAP	Mean Average Precision.
MSCOCO	Microsoft Common Objects in Contexts.
PASCAL	Pattern Analysis, Statistical Modelling and Computational Learning.
PASCALVOC	PASCAL Visual Object Classes.
RCNN	Region-Based Convolutional Neural Networks.
RGB	Red, Green, Blue.
RNN	Recurrent Neural Network.
SUN	Scene UNderstanding.
SVM	Support Vector Machine.
TEDM	Transformer-Encoder Detector Module.
TN	True Negative.
TP	True Positive.
TPR	True Positive Rate.
UAP	Universal Adversarial Perturbation.
VG	Visual Genome.
YOLO	You Look Only Once.

Publications

Alamri, Faisal, and Zunic,, Jovisa (2018). Edge Detection Based on Digital Shape Elongation Measure. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 19–27. Available at https://link.springer.com/book/10.1007/978-3-319-75193-1

The materials presented in chapters 3 and 4 have been published in:

Alamri, Faisal, and Pugeault, Nicolas (2019). Contextual Relabelling of Detected Objects. In *2019 Joint IEEE 9th International Conference on Development and Learning and EpigeneticRobotics (ICDL-EpiRob)*, pages 313–319. Available at https://ieeexplore.ieee.org/document/8850686

Alamri, Faisal, and Pugeault, Nicolas (2020). Improving Object Detection Performance Using Scene Contextual Constraints. *IEEE Transactions on Cognitive and Developmental Systems*. Available at https://ieeexplore.ieee.org/document/9137382

The materials presented in chapters 5 and 6 have been published in:

Alamri, Faisal, Kalkan, Sinan and Pugeault, Nicolas (2020). Transformer-Encoder Detector Module: Using Context to Improve Robustness to Adversarial Attacks on Object Detection. In *25th International Conference on Pattern Recognition, Milan, Italy, 2020*. Available at https://arxiv.org/pdf/2011.06978.pdf

Chapter 1

INTRODUCTION

Computer vision is one of the fields in computer science, which has been improving and enhancing very rapidly in the last decades. Computer vision can be defined as the field where computers are trained and monitored to interpret and understand the contents of digital images, including videos. Forsyth and Ponce (2002) also states that computer vision is "*extracting descriptions of the world from pictures or sequences of pictures*". Computer vision concerns the process of recognising faces, digits, objects and lots of other real-world items. It is applied and employed in several industries to improve human interactions, consumer experience, and even reduce cost and enhance security levels. For example, it can be used in the fields of retail, security, healthcare, and agriculture (Gundimeda et al., 2019; Shen and Zhu, 2013; Apruzzese et al., 2018; Chen, 2019; Tian et al., 2020). Such applications use computer vision to understand, analysis and/or process images for several purposes. However, recognising objects in a visual scene is an effortless task for humans, one that has challenged computer vision since its inception. The advent of deep neural network approaches over the last decades has delivered major improvements on this task on all benchmarks, although some difficulties remain Zou et al. (2019).

One notable weakness of Deep Neural Network (DNN) approaches to vision, first demonstrated by Szegedy et al. (2014b), is that perturbations of very small amplitude, barely visible to the human eye, can be found that lead to a major decrease in the neural networks' labelling accuracy—so called *adversarial attacks*. This seminal finding was confirmed by several studies, which showed that such attacks could be designed to be independent on the specific neural network or even neural architecture employed (Moosavi-Dezfooli et al., 2016), and that such attack patterns could be data independent (Mopuri et al., 2017). Looking at Figure 1.1, as human both images (left and right) may appear the same to us, no major visual differences can be observed, However, when inputted into a DNN classifier, it classifies them differently. On the left, the original image is passed into a DNN classifier, which classifies the presented object correctly as a `bear`, whereas when perturbation is added, as on the right image, it fails and misclassifies it as a `wolf`. This is how adversarial attacks can impact the performance of DNN approaches.

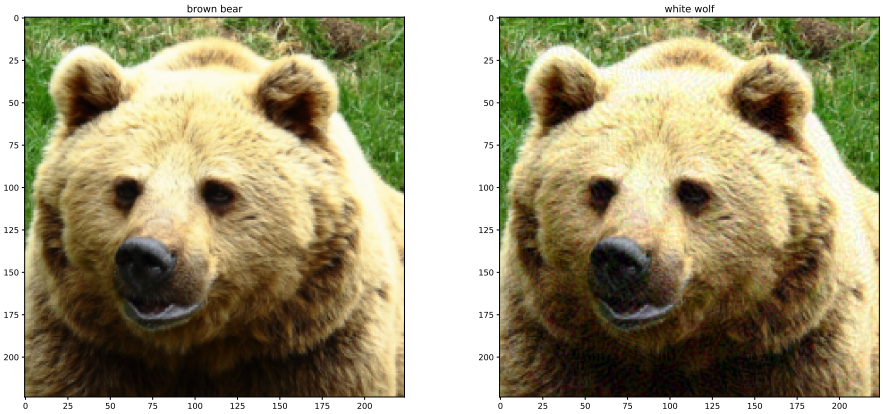


Figure 1.1.: Example of an Adversarial Attack

Context is defined by CambridgeDictionary (2020) as "*the situation within which something exists or happens, and that can help explain it*", which in terms of digital images, can mean the situation where objects are presented. Context, due to the rich information it carries, is suggested to address and mitigate the impact of adversarial attacks on computer vision systems. Imagine there are an *office desktop*, a *chair*, a *PC*, a *monitor* and a *keyboard*, then we ask the question "*what is the missing object in this scene?*". Some may say a *cup*, or a *pen*, but when looking deeply and understanding the context, you are likely to say the missing object is a *mouse* (Refer to Figure 1.2 for this imagined scene: Figure 1.2a shows when the mouse is missing, whereas Figure 1.2b shows when the mouse is presented.[1] Such guesses are made even without seeing or gazing at the scene due to leveraging contextual information. This shows how context carries rich information about visual scenes, upon this knowledge, contextual is believed to improve the performance of DNN models on both natural and perturbed images, as proposed in this thesis.

1.1. Motivation

As said, due to the importance of computer vision, it seems essential to improve computer vision performance. Detection, which is one of the most important tasks, helps to tell *where* and *what* objects are seen in digital images. Such a task is applied in autonomous vehicles, and security purposes, thus accuracy and speed are a need. One of the cues to improve detection performance is benefiting from context, which is the environments where objects are presented in images. Such a cue is seen to feed machine learning models with interesting, valuable and meaningful features that can be used to re-rate and reconsider models predictions. Therefore, deep investigations and explorations about scene contextual knowledge are aimed to be achieved in this thesis. This is expected to improve the development of machine learning models concerning the task of object detection. Detection type, studied in this work, is to detect and localise multiple objects within digital images; thus, contextual information is seen to help to enhance the performance, meeting the following thesis objectives.

[1] The Figure is moved to the next page to enable the reader to imagine before seeing the scene.

(a) Missing Object (b) Object is Added

Figure 1.2.: What is The Missing Object?

1.2. Objectives

This work concerns the importance of contextual information presented among objects in digital scenes. Therefore, this thesis aims to achieve the following objectives:

- Exploring and reviewing the state-of-the-art detection methods: Understanding how detectors work, and what can be done to improve them can help to plan for enhancement in their performances.
- Studying and understanding contextual information in digital images: before a task is attempted, it must be researched, read and understood; thus a further improvement can be accomplished when limitations and possible extensions are explored and fully observed.
- Developing models that leverage contextual constraints explicitly and implicitly: this is aimed to examine the effectiveness and the capability of contextual models in elevating detection performance.
- Investigating the following concerns: setting some questions to be answered is seen to help to keep the project on track. This can determine the milestones and plan further actions accordingly.
  - Q1: Do contextual constraints improve detection performance? If yes, to what extent do semantic, spatial and scale relationships enhance the detection performance?
  - Q2: Can contextual information be used to relabel and correct false detections?
  - Q3: Can the contextual models improve detection when perturbations are added to images?

1.3. Outline

The organisation of this thesis is made according to the objectives; thus, they can be met and achieved in a sequential manner as possible. Therefore, this thesis is organised as follows, as presented in Figure 1.3. Chapter 2 reviews the state-of-the-art detection methods. A brief description of the emergence of detection models and their types are also stated to familiarise myself and readers with the backbone of the proposed models in this work.

Chapter 3, as in the second objective, explores and provides the definitions and types of contextual information that can be extracted from digital scenes. A section about related work and approaches already developed is also presented. In this chapter, sixteen novel contextual relationships are proposed. Chapter 4, moreover, presents two novel contextual models that are built upon the use of the sixteen contextual relationships proposed, to rescoring and relabelling detectors predictions upon objects context, rather than only appearances. The models are named *Rescoring Model* and *Relabelling Model*, which extract contextual features from digital images explicitly following some mathematical equations that are also reported. Statistical and visualised results showing the performance of both models in comparison with two of the state-of-the-art detectors as examined on various datasets are presented in detail.

In addition, Chapter 5 illustrates further examinations of the proposed two models on perturbed images. This is to test the impact of such models, which object visual features can be affected and attacked. Two types of perturbations are applied, which are data-dependent, and data-independent, namely universal adversarial perturbation and fast feature fool.

Finally, in Chapter 6, a novel model, which implicitly extracts contextual features from images, is proposed. This model is built upon the success of a machine learning approach that is initially developed for Natural Language Processing (NLP) task, but is adapted due to its great performance and architecture. The proposed model, called *Transformer-Encoder Detector Module (TEDM)*, is evaluated on both natural images (no perturbations added), and perturbed images, and compared with Faster RCNN detector and the *Relabelling Model*, which is previously proposed.

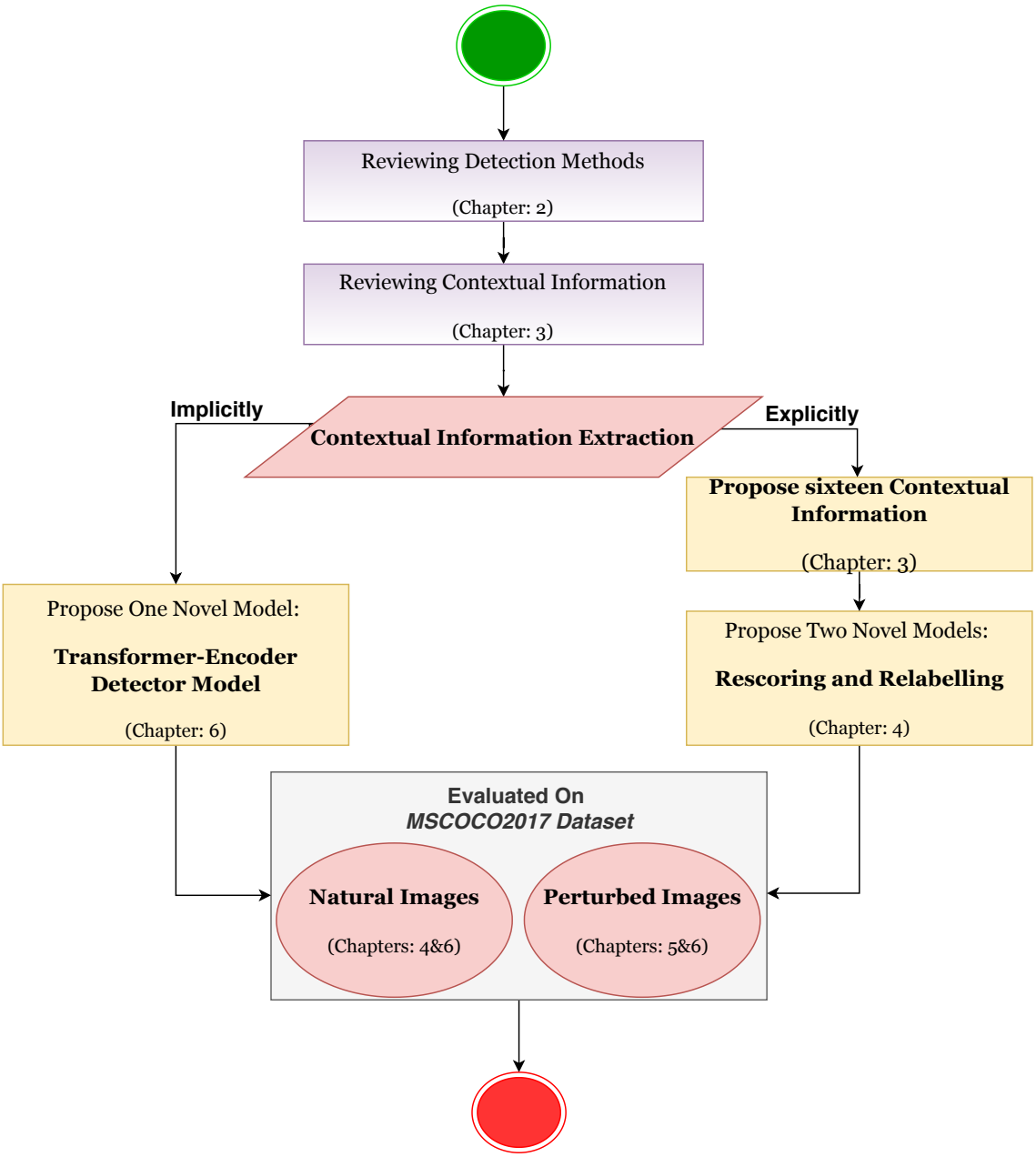


Figure 1.3.: Outline of This Thesis: Green and red circles represent where the objectives of this thesis are determined and achieved respectively.

In short, this thesis contributes to proposing the following:

- Sixteen contextual relationships, which are determined after exploring and studying the environments of objects in both MSCOCO 2017 and PASCAL 2012 datasets.
- A contextual model, which explicitly extracts contextual information from digital scenes, that re-rates detectors predictions, and determines the out-of-context objects, named the *Rescoring Model*.
- Upon the success of the *Rescoring Model*, another contextual model, named, *Relabelling Model* is developed to relabel and rescore detectors predictions depending on the information encoded from detected objects' statistical property and their vicinity including intra-class and inter-class details.
- A contextual model that implicitly extracts features from images and built on the advantage of attention mechanism to re-rate and relabel predictions upon the knowledge observed from both visual and contextual features.
- Examining the effectiveness of the contextual models on perturbed images due to the loss of features that attacks can cause.

Chapter 2

OBJECT DETECTION AND RECOGNITION BACKGROUND

2.1. Overview

Computer vision is one of the fastest growing fields in computer science, which includes object detection and recognition. Object detection models and techniques, without a doubt, have been enhanced very quickly, especially in the last two decades. Since 1998, the number of publications that concern object detection and recognition has been increasing fast, where many algorithms and models are proposed (Zou et al., 2019).

Object detection is different from object segmentation and object classification. Detection mainly aims to find out where objects are located in digital images, whereas classification concerns to find what objects are in the image. Object recognition is seen to include both detection and classification, thus it outputs both the *what* and the *where* objects are in images. Object segmentation is similar to detection, but it aims to determine what pixels in the image belong to what object. Figure 2.1 shows an illustration presenting the differences between those types.[1] The top image shows the original image, which contains a red `bus`. Below this image, classification and detection outputs are shown, where we can notice the difference, as in the left (*i.e.*, classification), the output is the content of this image (*i.e.*, a `bus`), whereas, on the right, the `bus` is detected and located in the image. As this is a one-object detection, thus, only the `bus` is located. The reader is referred to Section 2.2, for further information about the difference between one-object and multiple-object detection. In the last row, the difference between segmentation and recognition is also reported. In segmentation output, as its name suggests, all pixels belong to the `bus` are highlighted, which can be processed further as a segment of the image. Finally, recognition output presents the name of the object (*i.e.*, `bus`) and its location.

[1] Image taken from MSCOCO http://cocodataset.org and labelled by the online tool *LableMe* http://labelme.csail.mit.edu

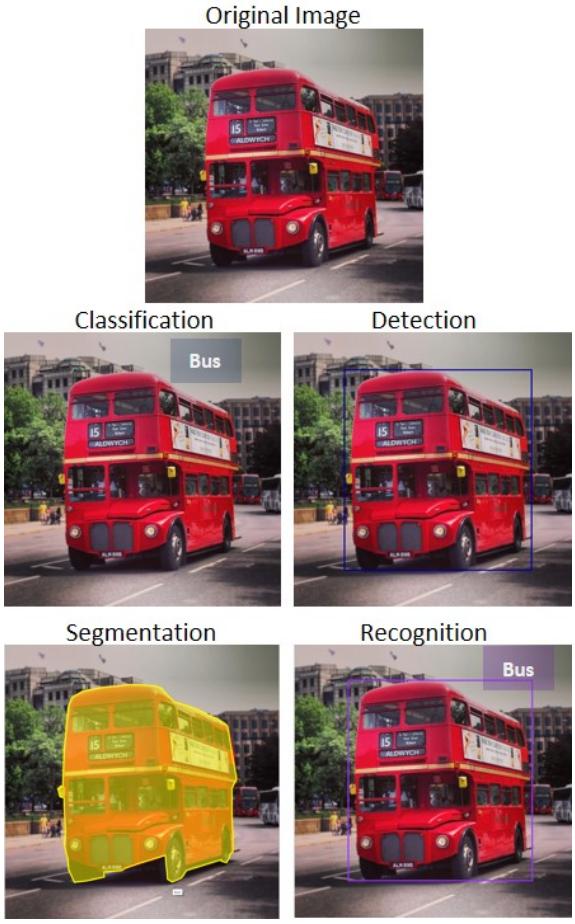


Figure 2.1.: Differences Between Classification, Detection, Segmentation and Recognition.

Location of objects in images, as seen in Figure 2.1, can be determined by the use of bounding boxes. A bounding box is used to describe and locate the target location, in a rectangular box, which can be determined by the x and y-axis coordinates. This can be presented in two formats: i) $[x_{min}, x_{max}, y_{min}, y_{max}]$, where x_{min} and x_{max} are the minimum and maximum of the x coordinates, and y_{min} and y_{max} are the minimum and maximum of the y coordinates, ii) $[x, y, w, h]$, where x is the x-coordinate, y is the y-coordinate, w and h are the width and height of the rectangle. Both formats can easily be exchangeable, as shown in Equations 2.1 and 2.2.

$$w = x_{max} - x_{min} \tag{2.1}$$

$$h = y_{max} - y_{min} \tag{2.2}$$

(a) Single Object Detection (b) Object Category Detection

Figure 2.2.: Types of Object Detection

2.2. Object Detection

One of the fundamental problems in the field of computer vision is object detection. It has been widely studied and improving. Object detection aims to determine whether an object from some predefined classes is present in given images. If objects are present, then object detectors are to output the spatial location (*e.g.*, bounding box), a confidence score (*i.e.*, how confident the detector is) and the predicted label. Object detection has been used in a variety of applications such as security, robot vision, autonomous driving and scene captioning (Liu et al., 2018a).

Object detection can be grouped into two categories, upon the aim of each type. First, *single object detection* that aims to detect an instance of a specific object class. However, the second type, known as *multiple object detection or generic-object detection*, which is considered to be more complex, as detectors of this type aim to detect multiple objects of the predefined classes (Liu et al., 2018a). Figure 2.2 shows the difference between both types. Figure 2.2a shows different instances of the same object category, whereas, in Figure 2.2b, it is shown different categories with two instance for each category, which is meant to present the latter detection type. Multiple object detection is also known as object-category detection, which is the main focus of this work.

2.2.1. Detection Methods

Since the 1990s, there has been an improvement in image feature descriptors and detectors. Features, here, can be any points of interest in the image, such as corners, colours and texture. Such great advances were made due to the developments of some feature descriptors and detectors. Detectors can be grouped into two categories, which are i) *traditional object detectors* (before 2012), and ii) *deep-learning-based object detectors* (since 2012). Figure 2.3 presents the timeline of some of the well-known detectors, where AlexNet is distinguished as it is one of the earliest deep-learning-based detectors, as it is considered to be the emergence of the second group of detectors. The differences between both types and how each works are presented in detail below.

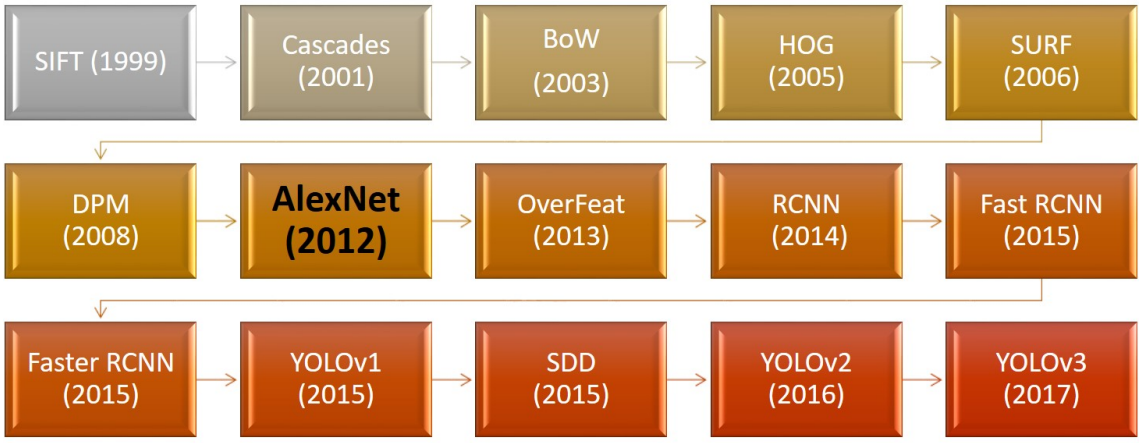


Figure 2.3.: Object Detectors Timeline

Traditional Object Detection

Object detectors 20 years ago were not as accurate and fast as they are nowadays; thus, they are called the traditional object detectors. Some of those detectors are Scale-Invariant Feature Transform (SIFT), which is proposed by Lowe (1999). It has been well-known and used in the field of computer vision, as it has been cited up to 19323 times.[2]

In 2001, the first facial detector was proposed by Viola and Jones (2001). It is named as 'Viola-Jones Detector', as on the authors' names. VJ detector uses a sliding window approach to go through the possible location where the face can be detected. VJ uses multi-stage detection, which is known as cascades detection, which was first proposed in this detector.

In 2005, Dalal and Triggs (2005) proposed Histogram of Oriented Gradients (HoG), motivated by detecting pedestrians. It detects not only large objects, but also small objects, as it re-scales the image multiple times; thus, it is scale-invariant. HoG has been applied in different studies such as (Malisiewicz et al., 2011). A year later, Speed-Up Robust Features (SURF) was proposed by Bay et al. (2006), and as its name suggests, introduced to boost the speed of the interest region extraction and/or description, as it is faster compared to SIFT (Luo and Oubong, 2009).

In 2008, Deformable Part-Based Model (DPM) was proposed by Felzenszwalb et al. (2008), as an extension of the HoG detector. DPM is the winner of PASCAL VOC07, VOC08 and VOC09 detection challenges. It follows the "*divide and conquer*" detection technique as described by Zou et al. (2019). In other words, it divides the object to be detected into parts (*e.g.*, motorbike can be divided into wheels, body).

Moreover, traditional detectors mainly follow a series of processes in detecting objects, which are: 1) obtaining the object location (*e.g.*, using sliding window), 2) extracting features (*e.g.*, SIFT), 3) applying classifiers (*e.g.*, SVM Cortes and Vapnik (1995)), and 4) revealing object location based on the classifiers outputs (Liu et al., 2018a).

[2]Number of citations is according to Google Scholar, last accessed 20/07/2020.

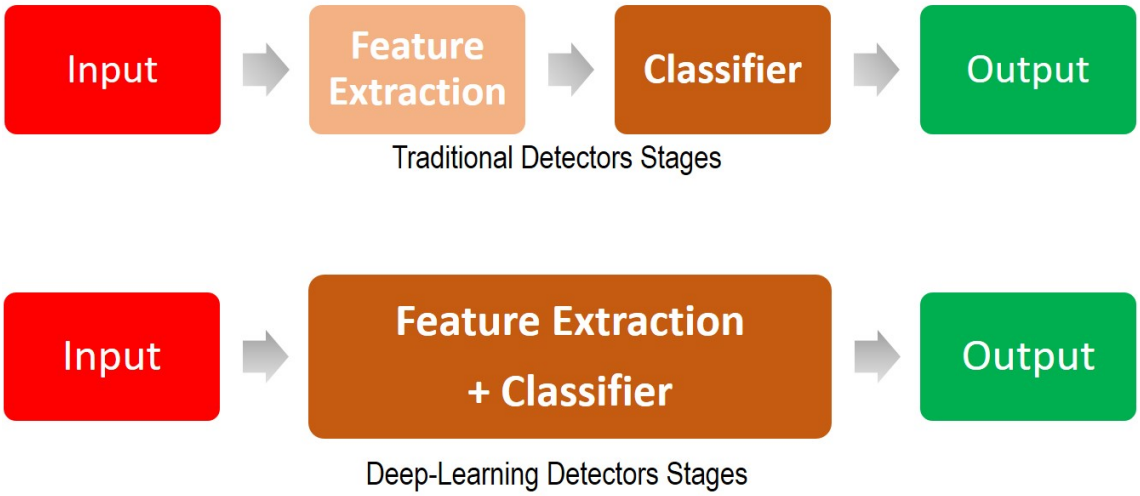


Figure 2.4.: Traditional detectors vs. Deep-learning detectors

As illustrated in Figure 2.4, It can be seen how the traditional detectors go through several stages to output the desired results, whereas, in comparison, deep-learning detectors shorten those stages to offer an end-to-end model, as detailed below.

Although traditional detectors produce reasonable results, they lack robustness to the variety of changes that may occur to the scene (*e.g.*, deformation and clutter), and they are time-consuming, which such disadvantages are seen to be overcome by the second type of detectors (described below) (Zhiqiang and Jun, 2017).

Deep-Learning Object Detection

Interestingly, a leap in the performance of object detection and recognition methods took place from 2012, when Convolutional Neural Networks (CNNs) were reintroduced. Girshick et al. (2013) claim that *"progress has been slow during 2010-2012, with small gains obtained by building ensemble systems and employing minor variants of successful methods"*.

CNNs were first modelled in 1998 (Lecun et al., 1998), but they were not widely used due to the computational power (*i.e.*, computers were not as fast as they are nowadays), and limited data (*i.e.*, number of datasets is improving, the more you feed a CNN detector, the better it is assumed to become).

Therefore, since 2010, due to the emergence in computers and datasets, CNNs have been the dominant in the computer vision tasks and the state-of-the-art detectors (Zhiqiang and Jun, 2017). Typical CNNs consist of some main layers; input layer, convolutional layer (Conv layer), pooling layer, fully-connect layer and an output layer, refer to Figure 2.5, for an illustration of a typical CNN structure. The input layer is where the image is inputted for further processing. This is then followed by a convolutional layer, which is the major block of the model. In this layer, filters (a.k.a. kernels), which are handcrafted, are applied

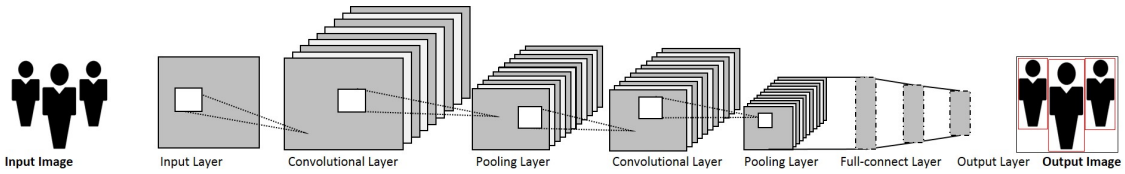


Figure 2.5.: Typical CNN Structure

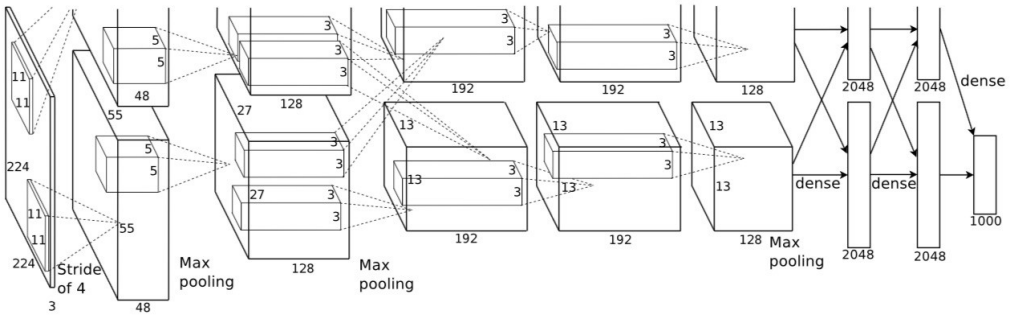


Figure 2.6.: AlexNet architecture, adapted from Krizhevsky et al. (2012)

to produce a feature map. A pooling layer is applied to downsample the size of the created feature maps. Two common pooling methods are used, which are i) average pooling and ii) max-pooling. They compute the average and the maximum value of each feature map, respectively. Outputs of the pooling layer are then flattened and passed into the fully connected layer, which is usually a simple, feed-forward neural network that drives the final classification decision.

In 2012, AlexNet, the winner of ILSVRC 2012 challenge, was introduced (Russakovsky et al., 2014). It contains eight learned layers, which are five convolutional layers and three fully-connected layers (Krizhevsky et al., 2012). AlexNet takes RGB images sized 256×256 as the input, meaning that all images inputted into this model need to be sized as 256×256. Random crops sized 227×227 are created from the inputted image to feed the first layer. The first layer consists of 96 convolutional Kernels sized $11 \times 11 \times 3$ used to extract interesting features. Feature maps created are then passed into an overlapping max-pooling operation that downsamples the width and height of the maps. Similarly, in the second Covn layer consists of 256 5×5 kernels, following by overlapping max-pooling as well. However, the third, fourth and fifth Conv layers are directly connected containing 384 3×3 kernels, 384 3×3 kernels, and 256 3×3 kernels, respectively. At the end of the fifth layer, an overlapping max-pooling is applied again. The output is then fed into two fully connected layers. Finally, outputs are passed into a softmax classifier with 1000 class labels. Figure 2.6 shows an illustration of AlexNet architecture. A noticeable improvement is seen in the field of computer vision since AlexNet was proposed; hence, CNNs have been widely used and enhanced.

CNN-based detectors can be categorised into two main groups, **1) Two-Stage Detection CNNs** and **2) One-Stage Detection CNNs** (Liu et al., 2018a). The former includes a pre-processing step where features are extracted and then passed into the pipeline. The latter is called a unified pipeline. As its name suggests, this group of CNNs has only one

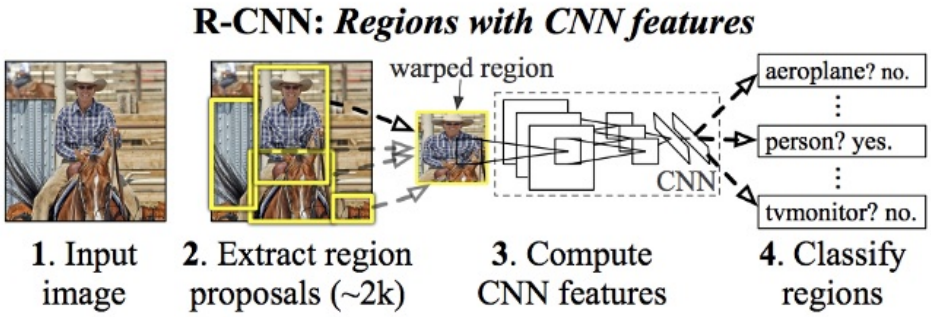


Figure 2.7.: RCNN Architecture, adapted from Girshick et al. (2013).

stage where feature extraction and prediction of classes are happening. Below, both types are explained in more details.

Two-Stage Deep-learning Detectors

Two-stage detectors, also known as region-based detectors, process detection in a "*coarse-to-fine*" manner, meaning that they process images in low-resolution then gradually increasing the resolution and propagating the results to the 'finer' image. In other words, the detection process in this type of detectors occurs in two steps. First, the detector proposes a number of regions of interest (RoIs) using some algorithms (*e.g.*, selective search), where those regions are then fed into the CNN networks. Second, classifiers (*e.g.*, SVM) are applied to provide probabilities for each detected objects. A brief explanation of some detectors belong to this group is presented as follows.

- Region Convolutional Neural Network (RCNN):

RCNN was proposed in 2013 by Girshick et al. (2013). It processes the image by extracting a set of region proposals ($\approx$ 2000 regions) using selective search algorithm (van de Sande et al., 2011). Thus, instead of processing the entire image, only 2k regions are processed. Each region proposal, contrast to scanning window, is then warped into a square fixed size to be fed into the CNN model, where features are extracted. Linear Support Vector Machine (SVM) classifiers are then used to predict the presence of objects. Figure 2.7 shows an illustration of RCNN structure.

As illustrated in Figure 2.7, RCNN consists of four stages from inputting the image to outputting where and what the objects in the image are. First, an image is inputted, which is processed to generate the regions proposals. This process takes place with the help of selective search algorithm, which looks at the inputted image using windows with different sizes. Each window works to group all adjacent pixels by texture, colour, or intensity to identify potential objects. Second, those proposed regions are passed into the CNN-network that performs the feature extraction. Prior this stage, all regions are then wrapped as a 227×227 RGB image, which are then fed into the CNN-network, which is a pre-trained AlexNet (*i.e.*, consists of five convolutional layers and two fully connected layers). After features of each proposal are extracted, they are passed into the final stage.

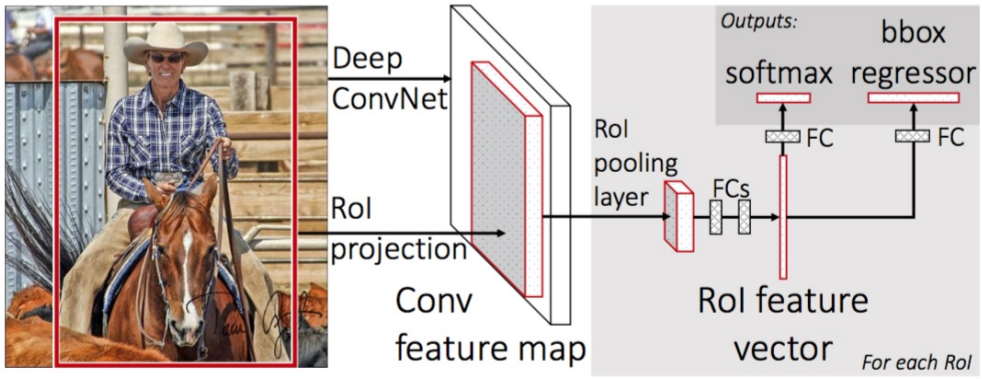


Figure 2.8.: Fast RCNN Architecture, adapted from Girshick (2015).

This stage is where every region is classified by a set of class-specific linear SVMs. Greedy non-maximum suppression is also applied to reject any regions with a high intersection-over-union (IoU) overlap, where the outputs are produced showing the labels, scores and positions of detected objects. SVM, the classifier used in this model, is a supervised machine learning model that uses classification algorithms for two-classes problems. It works by classifying inputted data by finding the best hyperplane (*i.e.*, decision boundary) that can separate all data points of class one from the other (Awad and Khanna, 2015).

Although RCNN shows a great improvement in detection performance, as it increased the mean Average Precision (mAP) on VOC07 from 33.7% (DPMv5 performance [3]) to 66.0%, it has some drawbacks, which are as follows. It uses selective search, which is a fixed algorithm; hence no learning is taking place at this stage. In addition, as mentioned earlier, it proposes about 2000 region proposals per image, which makes it too slow and cannot be used as a real-time application. Therefore, Fast RCNN is proposed, as explained below.

- Fast Region Convolutional Neural Network (Fast RCNN):

Girshick (2015) proposed Fast RCNN in 2015, which is about 25 times faster than RCNN, obtaining a higher performance in VOC07 (mAP = 66.9%). Fast RCNN enables images to be entirely fed into the CNN model for feature extraction, where Region of Interests (RoIs) are warped into a fixed size and then passed to a fully connected layer. Later, a softmax layer is used to predict objects, where boundary boxes are determined using bbox regressor. Fast RCNN is, therefore, different from RCNN as it uses only one network where features are extracted, classified, and bounding boxes are tightened, as presented in Figure 2.8.

- Faster Region Convolutional Neural Network (Faster RCNN):

Ren et al. (2015) propose Faster RCNN, which is 250 times faster than RCNN. Faster RCNN, the baseline detector used in this work, does not require region-extraction algorithms as in Fast RCNN, it instead uses a network that can be trained to take features to be inputted into a region-of-interest pooling layer. Hence, Faster RCNN feeds the entire

[3] Approach is proposed by Girshick et al., available on https://www.rossgirshick.info/latent/

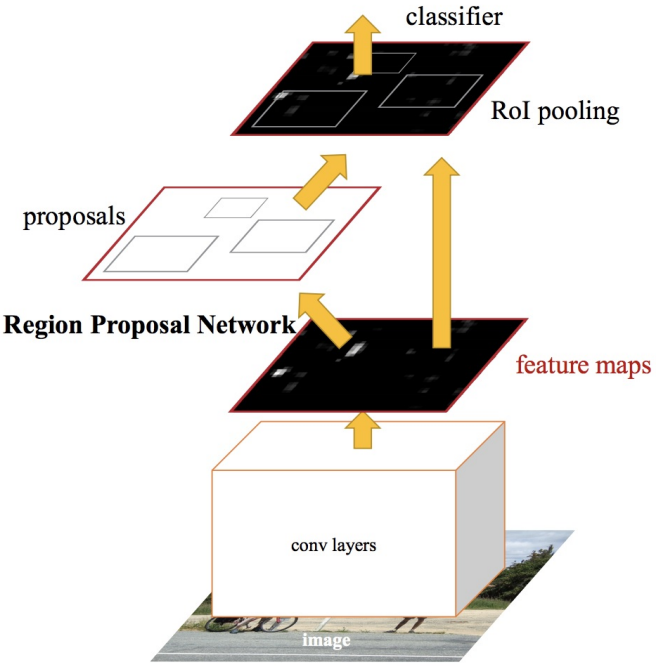


Figure 2.9.: Faster RCNN Architecture, adapted from Ren et al. (2015).

image into the CNN, where regions are extracted and then fed into the other layers (*i.e.*, ROI pooling, fully connected layers) for classification and bounding box regression. An illustration of Faster RCNN structure is presented in Figure 2.9.

In detail, Faster RCNN is slightly different from its descendants (*i.e.*, RCNN and Fast RCNN) due to its architecture and speed. It first takes the entire image as its input, which is then passed into a CNN (*i.e.*, VGG-16 is used (Simonyan and Zisserman, 2015)) to produce the feature maps. Features are then taken to a separate network, called Region Proposal Network (RPN). This is one of the contributions of this model, as it excludes the use of external algorithms (*i.e.*, selective search as in RCNN). RPN works to output interesting boxes/regions, which are then used for further processing. In other words, it ranks region boxes (*i.e.*, called anchors) and proposes the ones most likely containing objects. Anchors are just boxes with different sizes. In Faster RCNN, there are nine anchors, each three represent the scales of 128x128, 256x256, and 512x512, respectively, where each of the three has height-width ratios of 1:1, 1:2 and 2:1 respectively. Anchors are chosen to have this variety of scales due to the need of detecting different real-word objects (*i.e.*, humans, cars, etc.) which differ in sizes. Each anchor after scanning the image is examined by a classifier and a regressor, which check the presence of objects. Classifier, at this stage, only checks whether there is an object of not (*i.e.*, two-class probabilities), where the regressor predicts the four coordinates of a bounding box (*i.e.*, [x,y,w,h]). Outputs are then passed into the ROI pooling, which works to fix sizes of ROIs. Later, the outputs are fed into the final classifier and regressor, to obtain the class scores, labels and bounding box predictions.

In addition, Faster RCNN has been implemented in a variety of papers concerning the importance of contextual information such as Bell et al. (2015); Shrivastava and Gupta (2016); Zeng et al. (2016); Li et al. (2016); Hu et al. (2018); Liu et al. (2018b) as it is

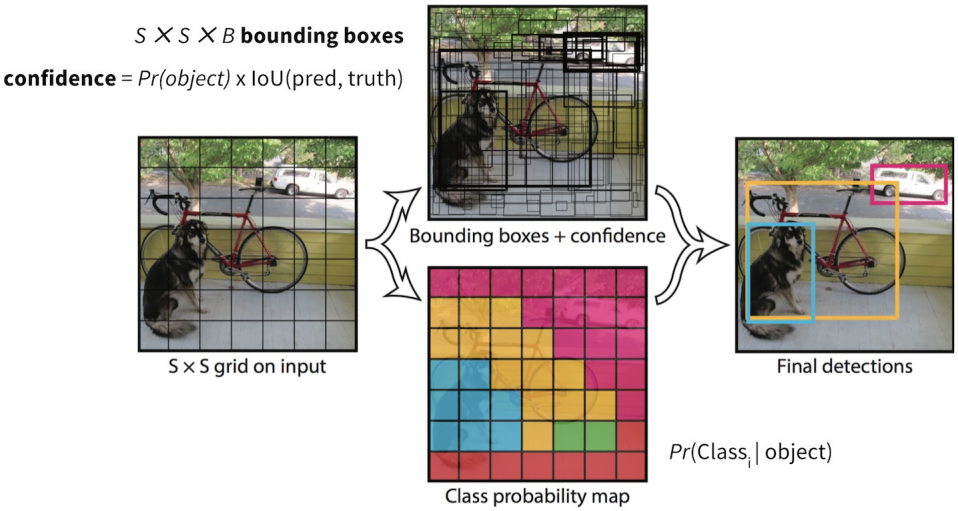


Figure 2.10.: YOLOv1 Architecture, adapted from Redmon et al. (2015)

still one of the state-of-the-art detection methods, and due to some of its advantages (*e.g.*, speed, accuracy), it is also used as the baseline detector for this work.

There are some other detectors that are not mentioned in this work such as SPPNet (He et al., 2014), RFCN (Dai et al., 2016), Mask RCNN (He et al., 2017) and Light Head RCNN (Li et al., 2017). Such detectors are not explained in this work, as they have not been used, where the detectors applied in this work have only been used as they are (*i.e.*, used as a black-box, no changes have been attempted).

One-Stage Deep-learning Detectors

One-stage detectors are also known as unified pipeline detectors. This group of CNNs has only one stage where feature extraction and prediction of classes are happening. Some of the detectors belong to this group are presented below.

- You Only Look Once (YOLO)

YOLO was proposed by Redmon et al. (2015) in 2015. It is considered as the first detector of this detection group (one-stage detection). It processes the input image by dividing it into regions then predicts bounding boxes and probabilities for those regions simultaneously, where those processes are taking place in one single network. YOLO is fast, and thus it can be used as real-time detector. It was tested on VOC07 obtaining mAP as 52.7%. Later two new and faster versions of YOLOs were introduced, which are YOLOv2 (Redmon and Farhadi, 2016) and YOLOv3 (Redmon and Farhadi, 2018). Detection performance on VOC07 has increased to 63.4% . Refer to Figure 2.10 for an illustration presenting YOLOv1 architecture.

As said, YOLO, which has 24 convolutional layers followed by two fully connected layers, is different from detectors stated earlier due to its architecture with processing the extraction, classification, and localisation of classes in one end-to-end model. YOLO takes the entire image as the input, which then splits the image in grids. For each grid, the model predicts

the bounding boxes (*i.e.*, [x,y,w,h]) and confidence scores reflecting the presence of an object. Non-maximal suppression is then applied to remove any duplications of detections (*i.e.*, those with lower confidence).

Single Shot MuliBox Detector (SSD) (Liu et al., 2015), DetectorNet (Szegedy et al., 2013), OverFeat (Sermanet et al., 2013), RetinaNet (Lin et al., 2017) are also examples of this group of detection.

In short, as Faster RCNN and YOLO are used though-out this work, only the comparison between them are reported. The main differences between both groups of detectors via a very deep comparison attempted by Zhao et al. (2018) and Zhiqiang and Jun (2017) are that Faster RCNN is seen to detect smaller objects (*i.e.*, even objects that are close to each other), which YOLO lacks due to the use of different sized anchors that Faster RCNN applies. YOLO, on the other hand, is considered faster than Faster RCNN due to the use of single-stage approach, whereas the performance of Faster RCNN outperforms YOLO on PASCAL VOC07, obtaining 73.2%, and 63.4% for Faster RCNN and YOLO, respectively as reported by (Sultana et al., 2019).

2.2.2. Main Challenges In Object Detection

Multiple object detection has been improving in the last decade due to the emergence of CNNs that mark a pivotal change in object detection history. However, the questions, to be addressed here, are *to what extent have new detection methods have improved the detection performance?*, and *what are the challenges that detection (both single and multiple object detectors) experience?*

Galleguillos and Belongie (2010) claim that the goal of object detection is to detect objects with high accuracy, speed and efficiency, thus they can be reliable and used in a real-time application, as possible. This goal is to be met, but there are some challenges that are still experienced, which are discussed below:

- **Inter-class and Intra-class Variations:**

Object detection, about 20 years ago, was an extremely complex problem. However, nowadays, they seem feasible due to the existence of deep-learning methods that overcome some of the detection obstacles. First, there is a huge number of object categories in real-world (*i.e.*, human, animals, vehicle, etc.); thus, detectors experience this as a challenge. Detectors were used to be trained for single object detection (*e.g.*, Dalal and Triggs (2005); Fergus et al. (2003)), whereas deep-learning detectors improve the multiple-object detection, as one can be trained for detecting more than one-object (See Section 2.2.1, for more information). Datasets such as PASCAL, MSCOCO, ImageNet (See Section 2.3 for more information) have a limited number of categories, which they have 20, 80 and 200 classes respectively. Although detectors are trained using any/all of those datasets, they will still miss other objects that are not presented in the datasets. Clearly, larger datasets with more object categories seems a need to address such an issue.

Figure 2.11.: Intraclass Variations

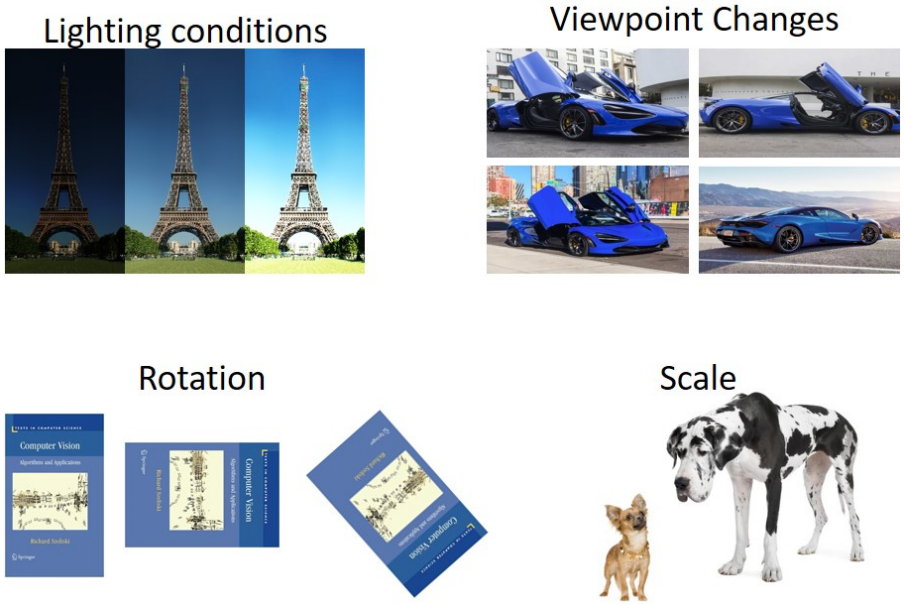


Figure 2.12.: Imaging Conditions Variations

In addition, even if detectors are trained to detect some pre-defined categories, intra-class variations are still an obstacle. This is because objects, in real-world, do appear in a wide range of sizes and aspect ratios. In detail, as shown in Figure 2.11, car, which is one object category, has different instances. In other words, objects (within the same category) may differ in appearance, shape, colour and texture. Anchor boxes with nine different sizes proposed in Faster RCNN use a small sliding window across the image's convolutional feature map to generate the candidate RoIs seem to address such an issue. It is, therefore, suggested to use larger datasets that include a large number of variations for each category to address such a challenge; thus the detectors can be trained more efficiently (Wang et al., 2016).

- **Imaging Conditions:**

Variations in images, caused by lighting conditions, differences in poses, scale, and changes in object orientation, can be another challenge for object detection, as illustrated in Figure 2.12.[4] It can be seen, as on the top left image, how lighting makes Eiffel Tower hard to be detected (some parts of it might be even hard for human to recognise).

[4]Images taken from Google Images, https://www.google.com/imghp?hl=en

Figure 2.13.: Out-of-Context Challenge, adapted from Instagram with permission from account @fransditaa

Moreover, in the top right image, we can also see that a car is captured from different views, where such changes in the viewpoint of objects can be another challenge. The disappearance of some important features of the objects may not be presented; thus, detectors fails to predict the correct objects. In real-world, objects are set in 3 dimensional, but in pictures (*i.e.*, used in this work), objects are in 2D. Therefore, if an object is taken from different views, occlusion of the objects' important points of interests could be an issue, which may lead to a decrease in the detector performance. Moreover, orientation, as in the bottom left image, can be another issue, as humans, we do not usually face a challenge in recognising objects even if they are rotated, because we have the mechanism of the understanding of orientation changes. However, systems may experience that as a challenge (Oñoro-Rubio et al., 2018).

Moreover, objects' sizes (*i.e.*, scale, as in the bottom right image) may mislead detectors when detecting objects that have similar sizes (*e.g.*, sheep and dogs). Detectors may also confuse in detecting objects that are changed in size. For example, when detectors are trained to detect an object of certain size, then tested on images presenting the same object but with different sizes, incorrect detection may be resulted (Wang et al., 2018).

- **Out of Contexts:**

Objects, in real-world, are likely to appear in such a spatial layout that they can be detected and localised. However, when objects are presented out-of-context, as in Figure 2.13, where the `cat` is clearly out-of-context. This can be hard for detectors to predict objects. This challenge is the focus of this PhD work, where it is deeply described and discussed throughout this thesis, particularly in Chapter 3.

Figure 2.14.: Interactions between different object classes and instances with the background

- **Multiple Instances Within One Image:**

Images may contain a few numbers of objects (either from the same category or different categories). Thanks to the developments of CNNs, which are now able to detect and recognise different objects within the same scene. However, this still seems as a challenge, especially when occlusions and background clutter occur. Objects can be covered by others, which then makes it hard for the detectors to predict the behind object due to the lack of appearance. This can be even harder when objects are not clearly isolated from the background, because background may change objects' features or be considered as a part of the objects. Figure 2.14 shows how occlusions may occur between objects that belong to the same category and with other categories in respect to the background as well. Some studies have widely been attempted concerning this challenge, such as Wang et al. (2017) who state that occlusion and deformation may mislead detectors. Therefore, they proposed an adversarial network that generates examples with occlusions and deformations to enable detectors to learn such invariances.

2.3. Datasets

Datasets are an essential element of object recognition research. Before I train an object detector, I will need data, to train detectors, *this is how important are datasets.*

As said, datasets have played a vital role in improving the state-of-the-art detection methods, because the more the detectors are trained, the better they are likely to become. Datasets are used in training object models and for testing the performance of classification, detection, and localisation algorithms. During the last decade, the numbers of images and categories in datasets are improving, leading to better detectors, but there are still smaller than the number of objects recognised by humans (Liu et al., 2018a).

There are several datasets that are widely used for image classification and detection. Datasets are also used for training, measuring and comparing the performance of models, some of which are listed as follows.

- Caltech 101:[5]

Caltech 101, as its name suggests, consists of 101 object categories. There are about 40-800 images per category, where most categories contain about 50 images. The total number of images is 9144. This dataset was collected and developed by Li Fei-Fei et al. (2004). Caltech101 has very low intra-class variation plus that it has only one object per an image.

- Caltech 256:[6]

In 2006, Caltech 256 was released. Caltech 256 was collected in a similar manner as in Caltech 101, proposed by Griffin et al. (2006). This dataset introduces a larger number of categories, containing 30607 images, where each category contains 80-827 images. Compared to Caltech101, this dataset seems to improve intra-class variation issue.

- LabelMe:[7]

LabelMe is not just a dataset, but also a web-based annotation tool that aims to provide an online platform for labelling objects in images to build image datasets. LabelMe is accessible by everyone to contribute to image annotation by visiting their online webpage.[7]. LabelMe is *"designed for object class recognition"* and *"learning about objects embedded in a scene"* (Russell et al., 2008).

- PASCAL:[8]

Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL) dataset is one of the early and common datasets used in the field of object categories detection. PASCAL is not only used in object detection, it is also used in multiple tasks such as image classification (Shetty, 2016), image segmentation (Pinheiro and Collobert, 2014), and action detection (Zhao et al., 2016).

It consists of 20 commonly found classes, which are 1) `aeroplane`, 2) `bicycle`, 3) `boat`, 4) `bottle`, 5) `bus`, 6) `car`, 7) `cat`, 8) `chair`, 9) `cow`, 10) `dining table`, 11) `dog`, 12) `horse`, 13) `motorbike`, 14) `person`, 15) `potted plant`, 16) `sheep`, 17) `train`, 18) `sofa`, 19) `TV`, and 20) `bird`. Images were collected from Flickr. PASCAL dataset was first released in 2003 consisting only four classes in 1578 images (Everingham et al., 2010). Two versions of PASCALVOC (Visual Object Classes) that are much well-known in the field of object detection are commonly used, which are VOC2007 and VOC2012. Both versions consist of 20 object categories, but differ in numbers of images. VOC2007 includes about 5k training images and 12k annotated objects, whereas the latter includes about 11k images with 27k annotated objects. PASCAL is gradually fallen out of fashion due to the presence of other datasets such as MSCOCO and ImageNet (presented later) as they have a larger number

[5]Caltech101 Webpage: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[6]Caltech256 Webpage: http://www.vision.caltech.edu/Image_Datasets/Caltech256/
[7]LabelMe Webpage: http://labelme.csail.mit.edu/Release3.0/
[8]PASCAL Webpage: http://host.robots.ox.ac.uk/pascal/VOC/

of images with more object categories (Zou et al., 2019). However, due to its wide use, and the presence of the common objects, it is used as a second dataset in some experiments attempted in this work to get the proposed models evaluated in more than one dataset.

- SUN:[9]

Scene UNderstanding (SUN) dataset was collected in 2010 to support both object and scene detection. It consists of annotated objects and scenes, which led to mitigate the issue of intra-class variation as well. Therefore, it is used in both object detection and scene description. SUN has 131,072 images within 908 scenes and 3819 classes (Xiao et al., 2010). Xiao et al. (2016) state that images were collected from Caltech 101 (Li Fei-Fei et al., 2004), Caltech 256 (Griffin et al., 2006), Tiny Images (Torralba et al., 2008), ImageNet (Deng et al., 2009) and Internet using online search engines. English sentences used for scene taxonomy were build by the help of WordNet (Fellbaum, 1998). Tiny Images dataset, however, has been retracted due to "*the presence of prejudicial images that hurt efforts to foster a culture of inclusivity in the computer vision community*", as reported by Torralba et al. (2008).[10] Dataset was withdrawn, as it was hard to guarantee all offensive content is removed because dataset was too large containing images that were too small (at 32x32 pixels). Such an offensive content has been mentioned and discussed in several studies such as (Prabhu and Birhane, 2020).

- ImageNet:[11]

ImageNet dataset is composed of over 14M images with over 200 classes (Russakovsky et al., 2014). ImageNet is the largest available dataset (Deng et al., 2009). There is a detection challenge called ImgageNet Large Scale Visual Recognition Challenge (ILSVRC), which was organised from 2010-2017.[12]

- MSCOCO:[13]

Microsoft Common Objects in Context (MSCOCO) is a well-known dataset in the area of context-based object detection. It was introduced by Microsoft in 2015. It consists of 80 common classes, which are listed in Table 2.1, where *common* is defined by Lin et al. (2014) as "*objects types that would be easily recognisable by 4-year-old*". It is composed of more than 120k images for training and validation and about 40k images for testing. It, on average, consists of 7.7 objects per image, which is a very high number of objects compared to ImageNet, which has only 3.0 objects per image or PASCAL, which consists of only 2.3. MSCOCO also contains more everyday scenes with common objects presented in their natural context, in contrasts of ImageNet, which tends to presents large and well image-centred objects. Therefore, MSCOCO seems to be the optimal dataset in a wide range of context-based studies such as Lin et al. (2014), which is, therefore, used in all the experiments in this thesis.

[9]SUN Webpage: https://groups.csail.mit.edu/vision/SUN/
[10]Tiny Images retraction announcement: https://groups.csail.mit.edu/vision/TinyImages/
[11]ImageNet Webpage: http://www.image-net.org/
[12]ILSVRC Webpage: http://image-net.org/challenges/LSVRC/
[13]MSCOCO Webpage: http://cocodataset.org/

Table 2.1.: MSCOCO 2017 dataset object classes.

Person	Carrot	Keyboard	Snowboard
Airplane	Cat	Kite	Spoon
Apple	Cell phone	Knife	Sports ball
Backpack	Chair	Laptop	Stop sign
Banana	Clock	Microwave	Suitcase
Baseball bat	Couch	Motorcycle	Surfboard
Baseball glove	Cow	Mouse	Teddy bear
Bear	Cup	Orange	Tennis racket
Bed	Dining table	Oven	Tie
Bench	Dog	Parking meter	Toaster
Bicycle	Donut	Pizza	Toilet
Bird	Elephant	Potted plant	Toothbrush
Boat	Fire hydrant	Refrigerator	Traffic light
Book	Fork	Remote	Train
Bottle	Frisbee	Sandwich	Truck
Bowl	Giraffe	Scissors	TV
Broccoli	Hair drier	Sheep	Umbrella
Bus	Handbag	Sink	Vase
Cake	Horse	Skateboard	Wine glass
Car	Hot dog	Skis	Zebra

- Visual Genome:[14]

Visual Genome (VG) is one of the largest dataset used for object detection, and mainly for scene description. It consists of more than 108k images, 3.8M object instance. It also concerns object-object relationships, as it contains more than 2.3M relationships (≈ 17.68 per image) such as *in, wear, near, next to, under, below, etc.* (Krishna et al., 2016). It has been used in several studies, such as scene description (Krause et al., 2016), visual relationship detection (Lu et al., 2016b), and scene graphs (Johnson et al., 2015; Anderson et al., 2016).

It is said that the more images with more objects that datasets have, the better it is, and then the more the detectors can be learned and better to become. Gauen et al. (2017) and Liu et al. (2018a) present a deep comparative study on those datasets, presenting the number of total images and categories, image size, the released date and more importantly, the average number of objects per image, where MSCOCO has the highest, thus, it is considered for this work.

2.4. Validation Approaches

There are several evaluation metrics, which can be used depending upon the tasks. It is very important that when evaluating the results of any proposed method, or comparing results with other existing methods is to use the same datasets, to ensure compatibility. Some of the validation approaches are as follows:

[14]Visual Genome Webpage: https://visualgenome.org/

2.4.1. Precision vs. Recall

Before exploring precision and recall, the terms True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) need to be explained, as they are needed for computing precision and recall. As shown in Table 2.2, TP is a correct detection, meaning that an object is correctly-detected, whereas, FP is false detection, which occurs when detector predicts an object where there is none. FN occurs when the detector mis-detecting an object, but TN occurs when detectors detect no object as there is none. For example, if an image contains a car and passed into a detector. If the detector detects the car, then it is TP, but if it does not, then it is FN. If the detector detects the car but not in the correct location, this can be FP. Finally, if the detector does not detect a car as there is none then it TN. See Section 2.4.2, for more information on how models predictions are determined as correct or not.

Table 2.2.: Difference between TP, TN, FP and FN.

Element	Ground-Truth (Real)	Predicted/Detected
TP	1	1
FP	0	1
TN	0	0
FN	1	0

In terms of precision and recall, they have been around since the 1950s. They were first formulated to evaluate document retrieval algorithms by counting the number of documents that were correctly returned against those who were not (Tatbul et al., 2018). Precision, the fraction of relevant instances among the retrieved instances, is the relation between TPs and the total number of TPs and FPs, as in Equation 2.3. It other words, it is the percentage of the model predictions being correct.

$$precision = \frac{TP}{(TP + FP)} \tag{2.3}$$

Recall is defined as the fraction of relevant instances that have been retrieved over the total amount of relevant instances, which is the relation between TPs to the total number of TPs and FNs, as in Equation 2.4.

$$Recall = \frac{TP}{(TP + FN)} \tag{2.4}$$

After Precision and Recall are computed, Average Precision (AP), which the most commonly used metric, can be computed. In case of multiple object detection. It is computed separately for each of the object categories, where then mean AP (mAP) can be calculated, which is the averaged of AP overall object categories.

Figure 2.15.: IoU Examples: The overlap ratio between the red and the green boxes is 0.7, 0.2 and 0 from left to right.

2.4.2. Intersection Over Union

In detection tasks, determining the percentage of precision (*e.g.*, the fraction of TPs) requires implementing another measure called Intersection over Union (IoU), which is calculated as in Equation 2.5. Figure 2.15 shows three examples where the overlap ratio between the red and green boxes varies. In the first image on the left, the red box is overlapping with the green one with above 70%, where in the middle, the overlap ratio is 28%. The right image shows no overlapping between the two boxes.

The model predictions can be determined as positive (correct detection) when they have a higher value than the predefined value, known as the threshold. This threshold may vary from one dataset challenge to another. For instance, in PASCAL when IoU between the predicted objects bbox and the ground truth, is equal or higher than 0.5[15], prediction is positive, but in MSCOCO, the averaged over ten IOUs (*i.e.*, [0.5: 0.05: 0.95]) is required.[16]

$$IoU = \frac{\text{area}(\text{Ref} \bigcap \text{obj})}{\text{area}(\text{Ref} \bigcup \text{obj})} \tag{2.5}$$

2.4.3. ROC Curve

Receiver Operating Characteristic Curve (ROC Curve) was first introduced in the 80's, to evaluate detection performance (Meur et al., 2008). It is a graphical plot used to illustrate the True Positive Rate (TPR) (Equation 2.6) against the False Positive Rate (FPT) (Equation 2.7) at various threshold values. It can help to determine the optimal threshold for the model examined. ROC leads to obtain the Area Under the Curve (AUC) score, which tells how the model performs, the closer this score is to 1, the better the model

[15] As stated in PASCAL web-page at: http://host.robots.ox.ac.uk/pascal/VOC/
[16] As stated in MSCOCO Detection Evaluation web-page at: http://cocodataset.org/

is.

$$TPR = \frac{TP}{(TP + FN)} \tag{2.6}$$

$$FPT = \frac{FP}{(FP + TN)} \tag{2.7}$$

2.4.4. F1 Score

F1 score is another evaluation matrix used to measure models performance. It is also known as F-measure or F-score. It considers both the precision and the recall to be computed, where F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0. F score is computed as in Equation 2.8.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.8}$$

2.5. Summary

In this chapter, some of the state-of-the-art detection methods were explained and compared with others. There are two types of detection, which are specific-object detection and genetic-object detection. Deep-learning object detectors, which mainly focus on the latter type of detection, can be grouped into two: 1) single-stage detectors, 2) two-stages detectors. As the aim of this work is to measure the impact and importance of contextual information, and how such a factor can enhance detectors performances, the latter type of detection is chosen due to the need of multiple objects. Two baseline detectors, one from each group, are used to examine the impact of contextual information (*i.e.*, Faster RCNN, YOLO). Experiments will take place with applying the chosen baseline detectors on two different datasets, which are PASCAL and MSCOCO. As said, PASCAL is chosen, because it contains 20 common objects, whereas MSCOCO is used as the main dataset for this work, because it contains large numbers of objects per image compared to other datasets. The performance of the proposed models, as experimented in Chapters 4, 5 and 6 will be evaluated using three evaluation matrices (*i.e.*, mAP, ROC, and F1).

Chapter 3

CONTEXTUAL INFORMATION

3.1. Overview

This chapter considers the importance of contextual information, presenting the definition and types of contextual information. Some of the related work concerning and studying contextual information are also reviewed in detail. Based on the knowledge obtained and observed from related work, sixteen relationships are proposed in this chapter, focusing on one-all objects relationships in digital images in terms of their sizes, distances between them and overlapping.

3.2. What Is Contextual Information?

Objects, in real-world, usually appear in particular environments with other objects known as the context. In other words, context is defined by CambridgeDictionary (2020) as "*the situation within which something exists or happens, and that can help explain it*", which in terms of digital images, can mean the situation where objects are presented. Therefore, the context in this work indicates the environments of objects in 2D digital images, where contextual information is the knowledge observed from such contexts. Contextual information, as said, plays an essential role in visual recognition for both human and computer vision systems. Figure 3.1a shows an object isolated from its context, which seems hard to be identified not only by systems but even by some humans, whereas when presented in context as in Figure 3.1b, it can be classified with less effort (*i.e.*, it is a `cup`). This example clearly illustrates the fact that contextual information carries rich information about visual scenes. It, in terms of object recognition, could be defined as cues captured from a scene that presents knowledge about objects locations, size and object-to-object relationships.

(a) Object Isolated

(b) Object In Context

Figure 3.1.: Importance of Contextual Information

Contextual information is defined as "*a statistical property of the world we live in and provides critical information to help us solve perceptual inference tasks faster and more accurately*" (Mottaghi et al., 2014). In this thesis, I will use the following definition:

Theorem 3.2.1 *Contextual information is any data obtained from an object's own statistical property and/or from its vicinity, including intra-class and inter-class details.*

It is said that contextual information seems a tool that can be used more with multiple objects, so that relationships among objects can be deeply understood (Desai et al., 2011). Mottaghi et al. (2014) also state that in digital images, objects with clear appearance (*e.g.*, large objects) are easy to detect, whereas, some small objects are harder. Therefore, contextual information can be a solution here as it provides stronger cues in detecting small, or occluded objects due to the context where those objects present (Park et al., 2010). In addition, contextual information is described as "*a natural way to improve detection*" (Barnea and Ben-Shahar, 2018). Liu et al. (2018a) state that using contextual information in the field of object detection can help to understand and explore object vicinity (*i.e.*, scene-level context) as applied in Bell et al. (2015); Zeng et al. (2016); Shrivastava and Gupta (2016), and also provides object-object relationships (*i.e.*, object-level context) as in Rabinovich et al. (2007); Gupta et al. (2015); Chen and Gupta (2017).

Moreover, contextual information has been widely studied in different areas, such as object localisation (Choi et al., 2012b), image segmentation (Gould et al., 2008), out-of-context detection (Choi et al., 2012a), image annotation (Lu et al., 2011), scene modeling (Bozcan and Kalkan, 2019), image understanding (Dai et al., 2017), cognitive robot (Celikkanat et al., 2016), and object detection as in this work, as discussed below.

3.3. Types of Contextual Information

Contextual information can be classified upon the sources of information that can be extracted from images. Biederman et al. (1982) state that there are five categories of object-environment dependencies, which are categorised as *"(i) interposition objects interrupt their background, (ii) support: objects often rest on surfaces, (iii) probability: objects tend to be found in some environments but not others, (iv) position: given an object in a scene,*

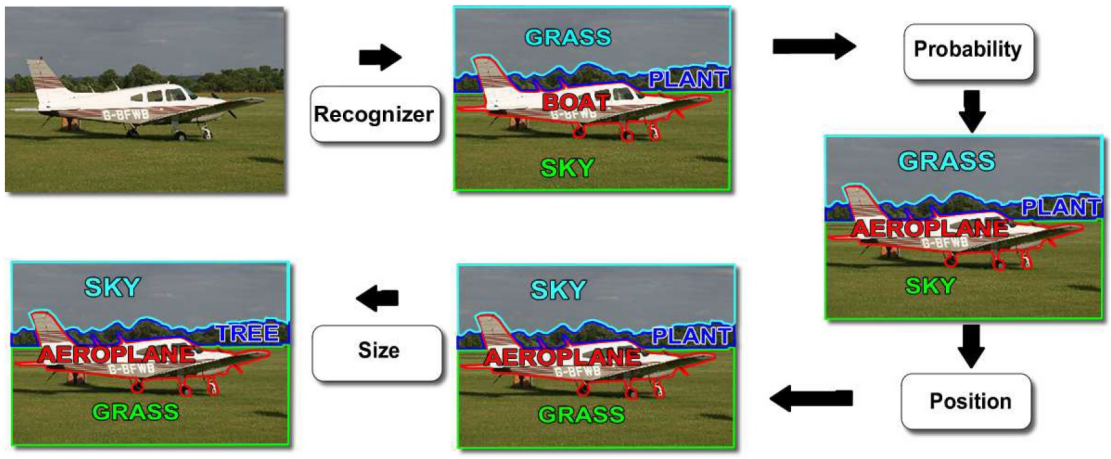


Figure 3.2.: Context categorisation, adapted from Galleguillos and Belongie (2010)

it is often found in some positions but not others, and (v) familiar size: objects have a limited set of sizes relative to other objects". Galleguillos and Belongie (2010) grouped those relationships into three main categories, which are (i) *Semantic (Probability), (ii) Spatial (interposition, support and position), and (iii) Scale (familiar size).*

Figure 3.2 shows an illustration of the categorisation and how each context is to refine the labelling. Divvala et al. (2009) support this categorisation of contextual information stating that context is widely used in the field of computer vision, but it is not well-defined; thus, it can be explored as any and all information that may influence the way a scene and the objects within it are perceived. In this work, I follow the same categories, as presented in Section 3.5.

3.4. Related Work

In this section, some of the state-of-the-art approaches that concern the development of contextual information and how detection can leverage contextual relationships among objects are presented.

In 2007, Rabinovich et al. (2007) published an article titled "objects in context", which focuses on the importance of context in improving categorisation accuracy. They propose a model that can be used as a post-processing step with any detectors (*i.e.*, at that time, only traditional detectors exit). This model is developed only upon the use of semantic relationship among objects in the scene. First, they input an image, and process image segmentation as a pre-processing step, followed by a categorisation step. Later, Conditional Random Field (CRF) framework is applied to incorporate semantic context and promote the agreement among the segments produced from the image segmentation process based on contextual relationships. It is reported that this model has improved categorisation accuracy when tested on PASCAL and MSRC datasets.[1]

[1]MSRC dataset can be accessed online at: https://www.microsoft.com/en-us/research/project/image-understanding/

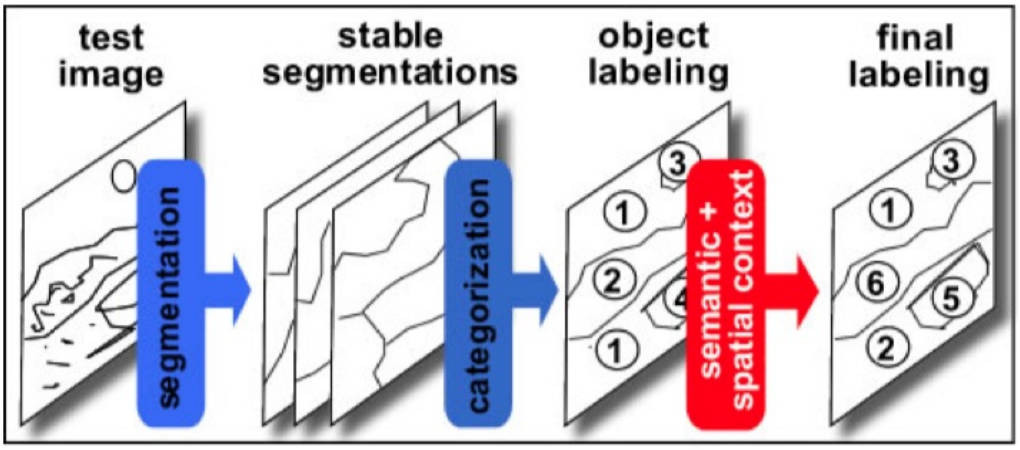


Figure 3.3.: CoLA Model, adapted from Galleguillos et al. (2008)

A year later, as an extension to the above work, Galleguillos et al. (2008) introduce an approach, named Co-occurrence, Location and Appearance (CoLA), that incorporates not only the semantic context but also the spatial context in addition to object appearance. They claim that such contextual knowledge helps in disambiguating object labels even if clutter, noise and variation in pose and illumination occur. In line with this, Bar and Ullman (1996) state spatial relations among objects decrease error rates in the recognition.

Similarly, CRF is applied to promote image segment agreement based on both semantic and spatial relevance. However, instead of concerning only semantic context, CoLA computes the frequency matrices for four pairwise relationships, which are *above, below, inside* and *around*. This model is evaluated on PASCAL VOC07, and MSRC resulting to an improvement in performance, which was reported as *"[this mode] improves accuracy in as many as half of the categories compared to using co-occurrence alone"*. Figure 3.3 presents an illustration of the processes involved in this model.

Choi et al. (2012b) propose a probabilistic contextual information model that captures the relationships among objects in digital images (*i.e.*, more than a hundred object categories from SUN dataset). This model is a probabilistic framework that uses the global image feature, relationships among object plus the outputs of the local detectors applied. The model was evaluated, resulting in an increase in object recognition performance, where the model also provides a coherent interpretation that may be used in scene understanding tasks. In detail, global features, called "gist" are integrated into the model. Gist is introduced by Oliva and Torralba (2001), described as that *"[it] can predict the presence or absence of objects and their locations without running an object detector"*. Contextual relationships are also encoded as: the co-occurrence statistics are encoded using a binary tree model where each node of the tree states whether the object is present or not. Spatial representations are then added in the tree model concerning the location and size of objects in pairwise relations. This model only considers the spatial layout between object classes, and not object instances, in a parent-child pairwise relationship. If the same object class occurs more than once in the image, the median location of all instances is then encoded. This dependency of relationships has its advantages, but also hard to be consistent in real-world images. For example, a keyboard is usually co-occurred with a monitor, but if

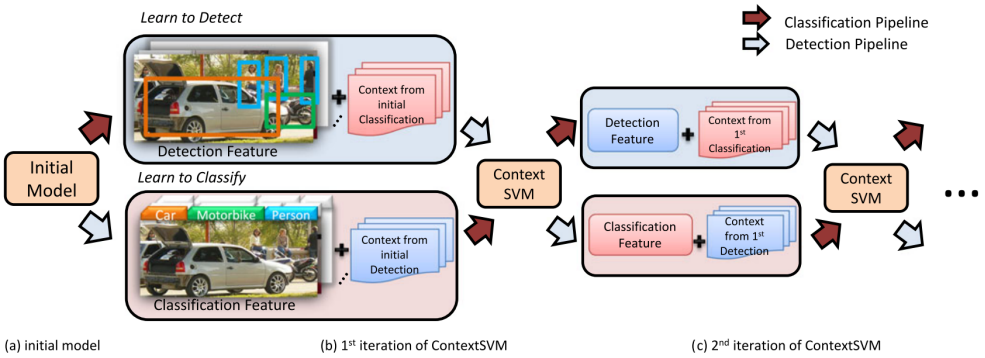


Figure 3.4.: Context-SVM Model, adapted from Chen et al. (2015)

the monitor only appears (no keyboard), the dependency is then lost and only assumed rather than computationally found. In this paper, spatial statistics dependency is built conditioned on the presence/absence of both child and parent objects.

Choi et al. (2012a) examine a number of contextual information sources and find out that physical support relationship among objects does provide the scene with deeper information compared to the semantic contextual relationship. It also helps to predict out-of-context objects, which appear in location where they are unlikely to be seen.

Moreover, Mottaghi et al. (2014) propose a novel deformable part-based model that uses both local and global image features. Local features are the information presented in the reference object vicinity, whereas global information is whether Object X is presented in the image or not. The proposed model is reported to provide significant improvement in object detection on PASCAL dataset. Contextualised Support Vector Machine (Context-SVM) is another model proposed by Chen et al. (2015). This model achieves the state-of-the-art performance as evaluated on PASCAL2007, PASCAL2010, and SUN09 at that time (*i.e.*, 2014). Context-SVM model aims to look at the image from a high-level task perspective, to focus on the contextualisation between object detection and classification. In other words, the proposed model first takes the input image, and obtains the detection outputs (*i.e.*, using some traditional detectors such as part-based model (Felzenszwalb et al., 2010)) and the classification outputs as well (*i.e.*, using Bag-of-Words (BoW) model (Fei-Fei and Perona, 2005)). Then, it benefits from both outputs to reduce error rates, as claimed that "*object detection and classification tasks can provide natural comprehensive context for each other without any external assistance*", as shown in Figure 3.4.

Cao et al. (2015) present a novel model, named High-Order Contextual Descriptor (HOOD), to compute the correlation strength among objects recognised within an image. Heterogeneous contextual information (*i.e.*, semantic, spatial and scale contexts) are jointly integrated into the proposed model. This model is to distinguish whether there is a positive or negative correlation and the dependency between objects. the descriptor is described using the probability of presence of objects classes and Bayes's rule, where the higher the value is, the more contextually consistency between objects. Such interaction between objects is constructed in a fully-connected object-level graph to represent the contextually consistent among them. HOOD is evaluated on both SUN09 and PASCAL 2007 showing that it outperforms the state-of-the-art context based object localisation methods as it also

detects out-of-context objects.

In terms of the application of deep-learning detectors in the use of contextual information, Chu and Cai (2016) claim that most of the RCNN-based detectors only concern the local features inside the bounding boxes, ignoring contextual information, which can lead to an incoherent understanding of the images; thus it is important to add such information to improve the performance. They propose a model that combines contextual information (local and global) and local appearance (*i.e.*, obtained from baseline detectors) as follows. First, in terms of the local features, semantic and spatial relationships are learnt from the statistical summary of the used dataset (*i.e.*, PASCAL VOC2007) in a pairwise manner. If objects A and B are overlapping, then six relations are defined (*i.e.*, inside, outside, up, down, left, and right), and if not overlapped, then objects are classified as far and five relations are defined (*i.e.*, up, down, left and right). Second, obtaining global features (*i.e.*, image-level contextual information) is done using a CNN model, which is trained for scene understanding tasks to provide ease in predicting the probability of how much some category is likely to occur (*i.e.*, if a lake is determined, then the likelihood of the presence of boats should be higher). Fully-connected conditional random field (CRF) is applied, where the graph nodes are the proposals produced by the baseline detector (*i.e.*, Faster RCNN). This method, as said, considers the contextual information in a pairwise relationship, but seems to significantly depend on the output statistics of the scene understanding model, because, as reported, if the model fails to gain the image-level scene, the refined scores can be "*worse*" than the baseline detector.

Inoue et al. (2017) propose a post-processing contextual information model that is applied on the top of Faster RCNN to rescore detector predictions. This model captures the semantic, spatial and scale contexts among objects, in the same manner as proposed in Choi et al. (2012b). Given the candidate windows produced by the baseline detector in a single image, the model first constructs a graph to represent the contextually consistent interactions. SVM is, then, applied to predict the new scores. This model is evaluated on MSCOCO2014 in comparison with Faster RCNN detector, where it shows a slight improvement in the mAP (33% compared to 32.2% as the performance for Faster RCNN).

Hu et al. (2018) propose "*the first*" fully end-to-end object detector that concerns objects relations. It is reported that there is no significant progress in the use of contextual information in the deep-learning era, and claimed to be one of the most first to address this issue due to the complexity in this application. As claimed, this complexity is raised due to the large number of different object categories, objects are at arbitrary image locations, numbers of objects varies in images, and objects have different scales. Therefore, they propose a model, called "*Object Relation Module*", motivated by the success attention modules in natural language processing field, that can help to learn object-object relations, to improve recognition and duplicate removal steps. Figure 3.5 shows the pipeline of this model, which are highlighted in dashed red. This model has the same number of dimension for both the input and the output, thus, it be applied within any network architecture. In terms of object detection, this model was applied on the basis of Faster RCNN. This model concerns the interaction between objects appearance features, which are extracted by the use of Faster RCNN and geometry features, which are pre-defined. The geometry

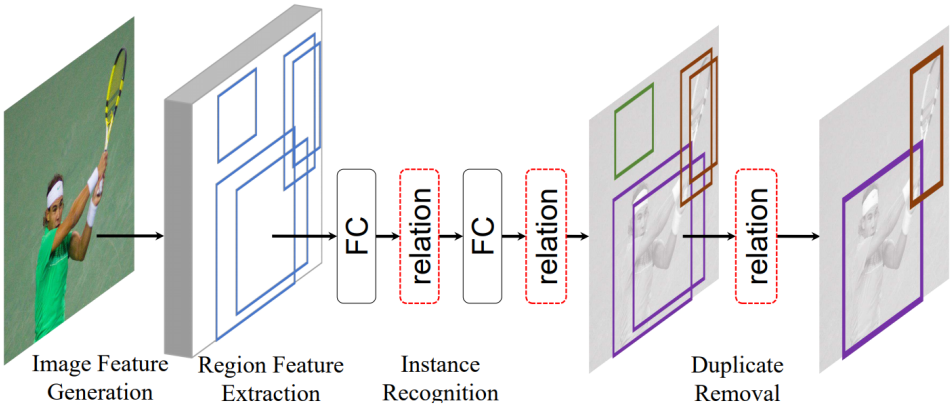


Figure 3.5.: Object Relation Module, adapted from Hu et al. (2018)

feature is 4- dimensional, which includes the pairwise relationships among objects, in terms of their scale and location.

Spatial and scale contexts have been also examined using probabilistic model, as proposed by Barnea and Ben-Shahar (2018). This model calculates the probability among objects in a belief propagation process to improve the detection performance by rescoring predictions upon the contextual information exploited from digital scenes. The proposed model improves the detection performance as evaluated on both PASCAL 2007 and MSCOCO 2014 datasets, where the baseline detector applied is Faster RCNN. It increases the mAP on COCO dataset from 66.5 to 66.9, and from 61.61 to 62.13 in PASCAL dataset.

Bozcan and Kalkan (2019) propose a model, named Contextualised Scene Modelling with Boltzmann Machines (COSMO), claiming to be the first to bring together objects, relations, and affordances in a highly-capable generative model using Boltzmann Machines (BMs) (Sejnowski, 1986). This model is applied in the field of scene modelling, but on the basis of using contextual information, due to its importance as it helps "*in resolving ambiguities, rectifying mispredictions, filtering irrelevant details*" (Bozcan and Kalkan, 2019). The proposed model can estimate relations between objects, find missing objects and irrelevant objects in the scene (*i.e.*, out-of-context). Eight spatial relations are considered (*i.e.*, *left, right, front, behind, on-top, under, above*, and *below*), and a set of verb actions. COSMO is evaluated on a dataset formed by the authors, which is composed of 6,976 scenes, half of which is sampled from the Visual Genome (VG) dataset and the other half from the SUNRGBD dataset to include a mixture of indoor and outdoor scenes. The model shows a great improvement in the mAP compared to Faster RCNN. It achieves 68% compared to Faster RCNN, which scores only 43%. However, the number of objects encountered into the model is seen very limited, which is considered a limitation for such a model.

Long Short-Term Memory (LSTM) networks have also been used as the baseline for encoding object contextual information, as in Li et al. (2019). The model proposed, used for captioning task, leverages object information within the scenes. It uses Faster RCNN for detection process, obtaining regions of interest. Only the first ten object detected with the highest confidence scores (upon the output of Faster RCNN) are considered, due to the limitation of LSTM. This model is evaluated on VG-COCO dataset (*i.e.*, an intersection

of Visual Genome dataset and MS COCO), which consists of 38,080 images for training, 2,489 images for validation and 2,476 for test. The model shows a better performance (*i.e.*, 4.67 points higher in mAP) compared to DenseCap, which is the baseline applied, proposed by (Johnson et al., 2015).

In short, contextual information, upon previous works, is seen and believed to improve the performance of several computer vision tasks including detection, classification and captioning. It can be implemented in a different method, pre-processing, end-to-end, and a post-processing upon detectors. In this work, it is applied as post-processing for most of the state-of-the-art detectors. The reader is referred to Chapter 4, for more information.

3.5. Proposed Contextual Relationships

In this work, I am following the division of contextual information proposed by Galleguillos and Belongie (2010). However, I have deeply studied those categorisation and propose sixteen contextual information relationships, which are as follows, summarised as in Figure 3.6.

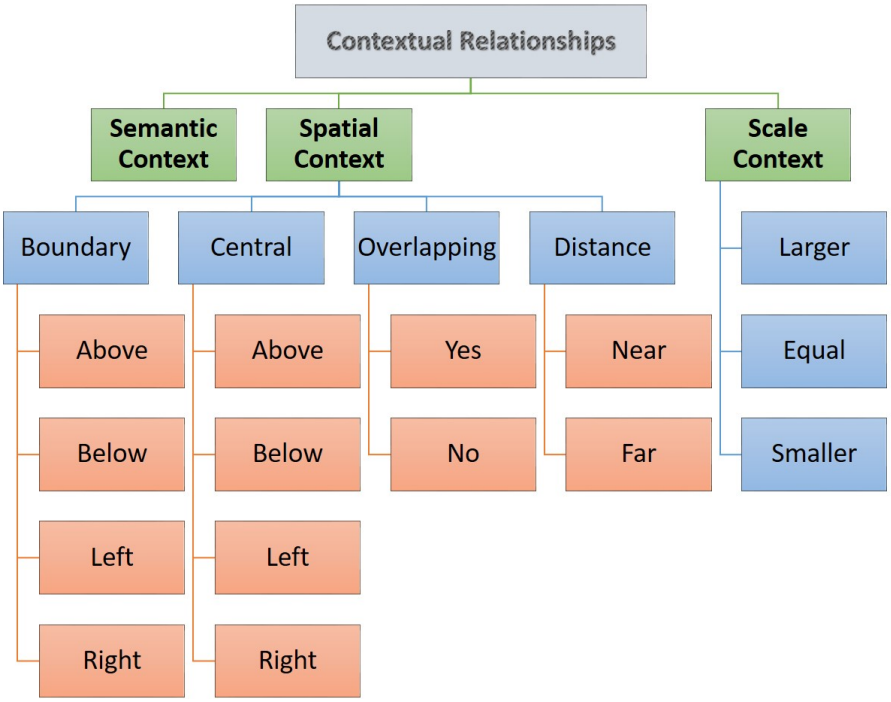
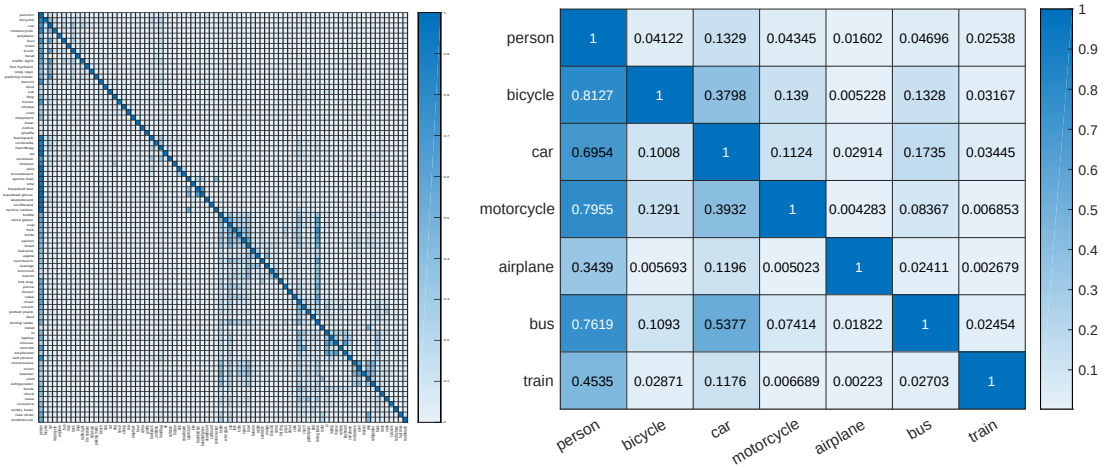


Figure 3.6.: The Proposed Sixteen Contextual Relationships

3.5.1. Category One: Semantic Context

Semantic context, also known as the co-occurrence statistics, concerns the co-occurrence among objects presented/detected in the digital scene. Semantic context is defined as *"the likelihood of an object to be found [presented] in some scenes but not others"* (Galleguillos and Belongie, 2010). For example, a car is more likely to be seen on the road rather than



(a) All Objects Co-occurrence Matrix

(b) Seven Objects Co-occurrence Matrix

Figure 3.7.: Co-occurrence Matrix: Figure b (right) is a portion of Figure a (left), shown for simplicity purpose.

in a bedroom, which then makes it more likely to co-occur with other objects presented in the road (*e.g.*, traffic lights). Such a type of contextual information helps to clearly find co-occurrence statistics among objects; thus, I can have a clear picture of objects that are more likely to appear in the same images. This can also help detectors to refine confidences when objects that have no co-occurrence were detected. For example, if a stapler is detected in the same image next to a keyboard and below a monitor, it would be more likely that stapler is a mouse, due to the high co-occurrence among those objects with respect the object appearance.

The semantic relationship has been widely studied and implemented in a variety of studies, showing an improvement in detection performance (Rabinovich et al., 2007; Ladicky et al., 2010; Mensink et al., 2014). Rabinovich et al. (2007) state that semantic contextual information is a very vital tool, as it helps to minimise the ambiguity in objects' visual appearance. Such a relationship is applied as a post-processing tool with local detectors showing that semantic *"greatly improves categorisation accuracy"*.

In this work, the semantic relationship is the first relationship applied. I deeply studied the training dataset and isolated images with a single object (*e.g.*, images with more than one object are used). A matrix, as shown in Figure 3.7, is built from the training images following the information presented in MSCOCO 2017 dataset annotations. If object X is presented with object Y, then co-occurrence among objects is positive. If both objects are within the same category (*i.e.*, human-alike: person) such as a man and a child are presented in the same image, co-occurrence will be considered positive as well. This information is used to obtain an overall view of the co-occurrence statistical in MSCOCO dataset, which helps to build the proposed models, as presented in Chapter 4. The reader is referred to Figure A.1 in Appendix A for the matrix shown the co-occurrence among all 80 MSCOCO objects.

3.5.2. Category Two: Spatial Context

Spatial context, as its name suggests, is defined as *"the likelihood of finding an object in some position and not others with respect to other objects in the scene"* (Galleguillos and Belongie, 2010). For example, a keyboard is more likely to be presented next to a monitor than next to a fork. Spatial information seems to provide deeper and more information about scenes than semantic as it concerns, not only the co-occurrence (*i.e.*, in an implicitly way) but also the location and relationships among objects (*e.g.*, a car is above the road). Although semantic relationship seems to provide a strong cue for disambiguating objects, adding more relations seems to improve detection even further. Spatial relationships have also been examined and studied in several researches.

According to Bar and Ullman (1996), who examined the consequences of pairwise spatial relations between objects, suggesting that encoding proper spatial relations among objects may decrease error rates in recognising objects. Many studies have included spatial context concerning only *above, below, left and right* relationships such as (Mottaghi et al., 2014). Others also added other types of spatial features such as *around and inside* (Galleguillos et al., 2008). Lu et al. (2016a), furthermore, add more relations such as *taller than, pushing, carrying.*

Choi et al. (2010) propose a contextual relationships model that is developed to leverage co-occurrences and spatial relationships among objects, using a graphical tree model, which is built to encode the dependencies among objects, as parent-child pairwise relationships. The outputs of this developed model are then combined with the outputs of the local detectors and the global image features. As reported, applying this model increases the performance of detection. In this work, as reported above, spatial information is seen as a vital cue in improving detection, thus, I am proposing four novel sub-groups of spatial relation, which are as follows.

- Boundary Spatial Relationships:

Boundary relationships consist of four types of relationships, which are *above, below, left* and *right.* As their name indicates, these relationships occurs when the boundary of the reference object is not attached or overlapped with other objects. In other words, it represents relationships between objects when there is a gap between objects' boundaries. For illustration, refer to Figure 3.8. For example, above relationship is considered when the bottom boundary of the object (shown in green) is above the reference object (in red); thus I say the green object is above the reference object. Refer to Table 3.1, for the mathematical equations of how such relations are computed. The reader is referred to Figures A.2 to A.5 in Appendix A for the statistical analysis for such relationships explored from the training dataset in MSCOCO 2017.

- Central Spatial Relationships:

Similarly, the second sup-group is named central spatial relationship, which is similar to the first sup-group. However, in those relationships, I do not just consider boundary; I also calculate the relationships upon the centres of the reference object and other objects.

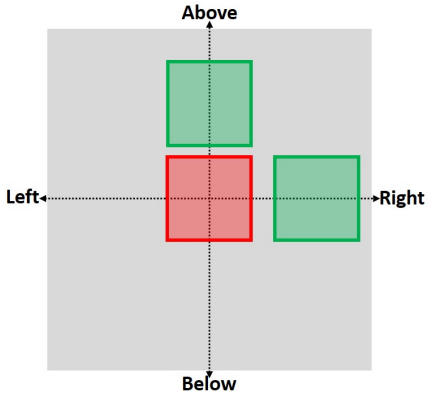


Figure 3.8.: Boundary Spatial Relationships: Red box represents the reference object, and green represents other objects.

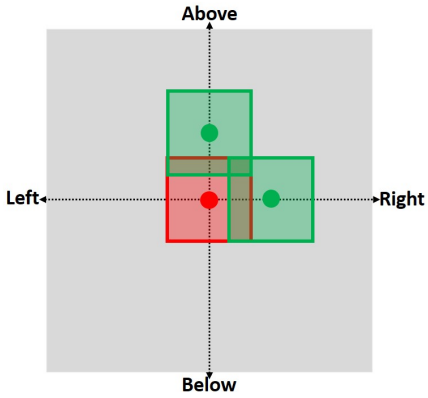


Figure 3.9.: Central Spatial Relationships: Red box represents the reference object, and green represents other objects.

Figure 3.9 shows an illustration of this relationship group. For example, above central relationship occurs when the centre of object X (shown in green) is above the centre of the reference object (shown in red) where the top boundary of object X is also above the top boundary the reference object. It can be said that this relationship includes several relationships, such as overlapping and size measures. I say yes, and this is what makes it unique as it focuses on the centres of objects but also encodes other relationships that are implicitly counted. Refer to Table 3.1 for the mathematical equations used for this sub-group. The reader is referred to Figures A.6 to A.9 in Appendix A for the statistical analysis for such relationships explored from the training dataset in MSCOCO 2017.

- Distance Relationships:

The third sub-group of spatial relationships concerns the distance between objects. Such contextual information is expected to enrich the context and provide deep knowledge about objects. Distance relationships consist of two types, which are *near* and *far*. An illustration of these relationships are presented in Figure 3.10, where mathematical equations used are in Table 3.1. The distance is measured to be near/far upon the diagonal of the reference object. If the boundary of object X (shown in green) is far by a distance that is larger than diagonal of the reference object, the relationship is considered far, and vice versa. The reader is referred to Figures A.10 and A.11 in Appendix A, for the statistical analysis

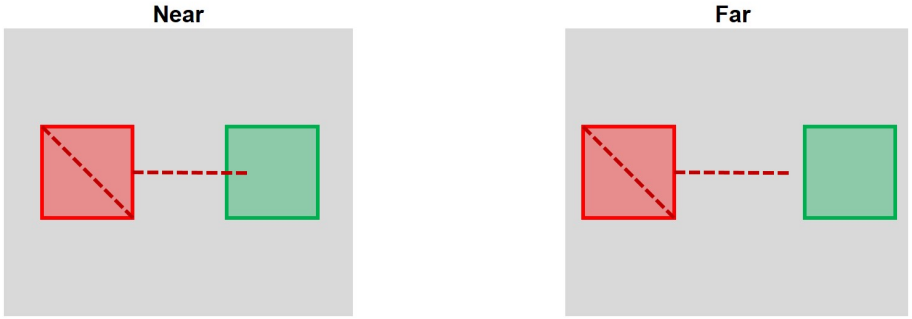


Figure 3.10.: Distance Spatial Relationships: Red box represents the reference object, and green represents other objects.

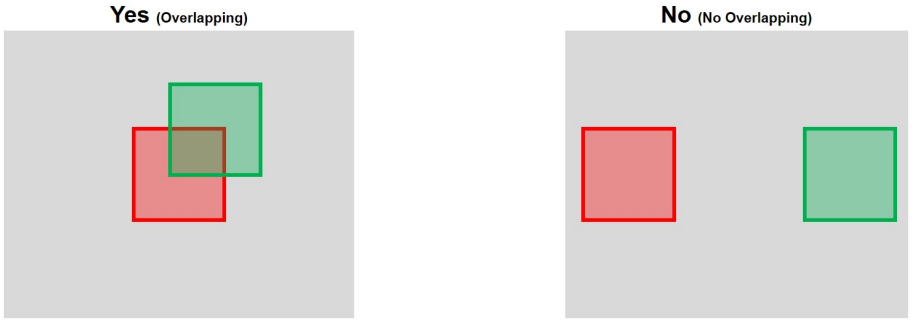


Figure 3.11.: Overlapping Spatial Relationships: Red box represents the reference object, and green represents other objects.

for such relationships reported from the training dataset in MSCOCO 2017.

- Overlapping Relationship:

Overlapping relationship, which is the fifth sub-group of spatial relationships, is to measure whether the reference object is overlapping with other objects or not; thus, it consists of two types, which are positive and negative overlapping. Overlapping ratio, as in Equation 2.5, is considered positive (*i.e.*, yes) when the Intersection over Union (IoU) value of objects is 0.5 or above. Figure 3.11 shows a presentation of such relationships, where equations used to measure this relationship is presented in Table 3.1. The reader is referred to Figures A.12 and A.13 in Appendix A, for the statistical analysis for such relationships upon the training dataset in MSCOCO 2017.

3.5.3. Category Three: Scale Context

Scale contextual information concerns the size of objects with respect to other objects in the same scene. It is proposed to measure objects sizes compared with the reference objects. It has been studied in many researches such as Galleguillos and Belongie (2010); Choi et al. (2012b); Mottaghi et al. (2014); Choi et al. (2012a); Barnea and Ben-Shahar (2018). The proposed scale relationships, in this work, are divided into three groups, which are *larger*, *smaller* and *equal*. Refer to Figure 3.12 for an illustration of those relationships, and Table 3.2 for equations used to measure scale contextual information among objects. Measurement, in this case, is counted only upon the diagonals of objects. If reference

Table 3.1.: Spatial relationships mathematical equations.

Boundary Relations	
Above	$(Ref_y + Ref_h) < Obj_y$
Below	$Ref_y > (Obj_y + Obj_h)$
Left	$(Ref_x + Ref_w) < Obj_x$
Right	$Ref_x > (Obj_x + Obj_w)$
Central Relations	
Above	$((Ref_y + Ref_h) \times 0.5) < ((Obj_y + Obj_h) \times 0.5)$ where $Ref_y < Obj_y$
Below	$((Ref_y + Ref_h) \times 0.5) > ((Obj_y + Obj_h) \times 0.5)$ where $(Ref_y + Ref_h) > (Obj_y + Obj_h)$
Left	$((Ref_x + Ref_w) \times 0.5) < ((Obj_x + Obj_w) \times 0.5)$ where $Ref_x < Obj_x$
Right	$((Ref_x + Ref_w) \times 0.5) > ((Obj_x + Obj_w) \times 0.5)$ where $(Ref_x + Ref_w) > (Obj_x + Obj_w)$
Distance	
Near by	$(Ref_x - (Obj_x + Obj_w)) < \sqrt{(Ref_w)^2 + (Ref_h)^2)}$
Far From	$(Ref_x - (Obj_x + Obj_w)) > \sqrt{(Ref_w)^2 + (Ref_h)^2)}$
Overlapping	
Yes	$Overlapping \geq 0.5$
No	$Overlapping < 0.5$

Table 3.2.: Scale relationships mathematical equations.

Scale Context	
Larger	$\sqrt{(Ref_w)^2 + (Ref_h)^2)} > \sqrt{(Obj_w)^2 + (Obj_h)^2)}$
Smaller	$\sqrt{(Ref_w)^2 + (Ref_h)^2)} < \sqrt{(Obj_w)^2 + (Obj_h)^2)}$
Equal	$\sqrt{(Ref_w)^2 + (Ref_h)^2)} = \sqrt{(Obj_w)^2 + (Obj_h)^2)}$

object diagonal is larger than object X diagonal, the relationship will be considered larger. The reader is referred to Figures A.14 to A.16 in Appendix A for the statistical analysis for such relationships upon the training dataset in MSCOCO 2017.

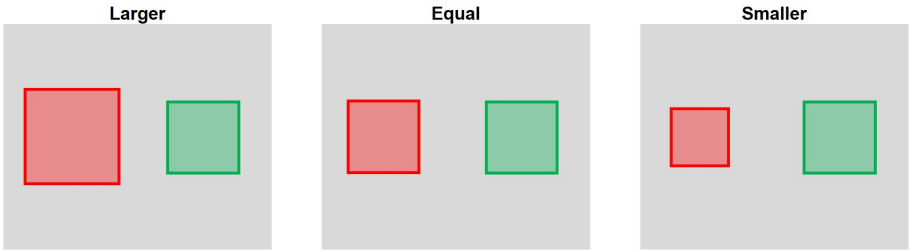


Figure 3.12.: Scale Relationships: Red box represents the reference object, and green represents other objects.

3.5.4. Statistical Results Obtained From Training Dataset

Table 3.3 presents some statistics obtained when examining the proposed sixteen relationships among five randomly chosen objects in MSCOCO 2017 training dataset. Six relationships statistics are presented, where the reader is referred Appendix A, for the

statistical analysis.

This table is to be read from the left to right, thus the relationships represent how objects in columns are related to objects in rows. For example, `person` has co-occurred with `bicycle` 0.04%, and noticed to be positioned above in 0.3% of images where both objects present. We can see that `person` has a high percentage in all relationships with the reported objects reported compared to others, as it is more likely to co-occur with all objects. In regards to scale larger relationship, we can observe that how `elephant` is related with other objects, as for `elephant-person` relationship 0.8% of the image both objects appear in, `elephant` is larger.

Table 3.3.: Statistical results obtained from MSCOCO 2017 training dataset, in percentages [%].

Co-Occurrence					
`Classes`	Person	Bicycle	Boat	Bench	Elephant
Person	1	0.0412	0.0308	0.0635	0.0139
Bicycle	0.8127	1	0.0376	0.0100	0.0061
Boat	0.1530	0.0409	1	0.0591	0.0052
Bench	0.07468	0.0595	0.0322	1	0.0205
Elephant	0.4137	0.0092	0.0072	0.0520	1
Boundary: Above					
`Classes`	Person	Bicycle	Boat	Bench	Elephant
Person	0.0911	0.3105	0.1973	0.2978	0.2014
Bicycle	0.0355	0.0974	0.0706	0.1862	0.0731
Boat	0.3478	0.0674	0.1738	0.6838	0.4318
bench	0.1190	0.1620	0.1515	0.1955	0.1629
Elephant	0.1034	0.1219	0.0901	0.0444	0.0739
Distance: Near					
`Classes`	Person	Bicycle	Boat	Bench	Elephant
Person	0.3522	0.3299	0.1706	0.3322	0.2895
Bicycle	0.3209	0.3694	0.2838	0.2839	0.1951
Boat	0.3347	0.1875	0.2684	0.2090	0.3181
bench	0.3237	0.2804	0.2888	0.3448	0.2444
Elephant	0.5377	0.6829	0.5681	0.6851	0.7142
Overlapping: Yes					
`Classes`	Person	Bicycle	Boat	Bench	Elephant
Person	0.4991	0.4765	0.6977	0.4847	0.8129
Bicycle	0.5225	0.4994	0.3544	0.5218	0.0714
Boat	0.3016	0.6455	0.4993	0.6242	0.3182
bench	0.5149	0.4770	0.3747	0.4992	0.8925
Elephant	0.1869	0	0.0681	0.1074	0.4993
Scale: Larger					
`Classes`	Person	Bicycle	Boat	Bench	Elephant
Person	0.4991	0.5225	0.3016	0.5149	0.1869
Bicycle	0.4765	0.3544	0.6455	0.4770	0
Boat	0.6977	0.3544	0.4993	0.3747	0.0681
bench	0.4847	0.5218	0.7612	0.4992	0.1074
Elephant	0.8129	0.0714	0.9565	0.8925	0.4993

3.6. Summary

In this section, the definition of contextual information and its types were stated. It has three main types, which are semantic, spatial and scale, concerning the co-occurrence, locations and sizes among objects, respectively. Following this division, I proposed sixteen contextual information that can explore object-object relationships, which are expected to improve the detection performance as studied in Chapter 4. As said earlier, this contextual knowledge provides information about objects vicinity and how they are presented in respect of other objects in the same scene. This, therefore, is expected to enrich detectors with more information about digital scenes, which is expected to be deeper than only the appearance features that can be obtained by detectors (See Chapter 2, for more information about detection). In the next chapter, the proposed sixteen relationships are applied as a pre-processing on two different detectors used as the baseline detectors, aiming to refine detected objects confidences in order to improve the detection performance and reduce false alarms.

Chapter 4

CONTEXTUAL RESCORING AND RELABELLING MODELS

4.1. Overview

Contextual information seems an effective property that minimises ambiguity in both human and computer vision systems. In this chapter, it is shown how the sixteen contextual relationships proposed in Chapter 3 are encoded, which exploit semantic, spatial and scale contexts, and how they help to improve the detection capacity and analysis various properties of the contextual object detection problem. Interactions among detected objects are well-studied, which is expected to lead to an increase in the detection performance, as to be experimented in this chapter, upon the use of of Faster RCNN and YOLO as the baseline detectors, and evaluated on both PASCAL and MSCOCO datasets,

Nowadays, CNNs-based detectors have been widely used in the field of object detection, as they are performing as the-state-of-art detectors due to their accuracy. However, contextual constraints are still not fully employed by such detectors, as they only depend on a region of the input that layers, ahead of classification layer, produce (Luo et al., 2017).

Therefore, they do not include contextual information that can be presented among regions. Previous models address such an issue by incorporating contextual information into the detection processes by adding further layers into their CNN networks (Bell et al., 2015; Li et al., 2016). However, as claimed by Barnea and Ben-Shahar (2018), those models even though they seem to improve the detection performance by including context, the contextual knowledge included is "*unclear*". They are still unable to *"reason about object relations in a manner invariant to viewpoint"*, where they require "*all meaningful relations between all groups of objects [to be observed] from all relevant viewpoints*" in the training dataset. Therefore, it seems a need to develop a model that can leverage contextual information, where contextual relationships among objects are clearly presented.

In this chapter, I am proposing two models that leverage contextual information encoded from digital images for rescoring object confidences and relabelling objects. The two proposed models are applied as a post-processing for most of the existing detectors for refining the confidences and associated categorical labels, without refining bounding boxes.

4.2. Rescoring

This section presents the effectiveness of the proposed *Rescoring Model* on the baseline detectors chosen. This is shown in several experiments, as presented below. Before I illustrate the experiments, I would like to present how the model works and how it is built.

4.2.1. Procedure of The Proposed Model

As shown in Figure 4.1, the model, in the training stage, first takes images from the training dataset, then passed them into the baseline detectors. Detectors, then, produce the prediction, which includes (for each detected object) the bounding boxes (*i.e.*, [x,y,w,h]), object label and the confidence score. The outputs (dimension is 6: 4 for bounding boxes, 1 for the label, and 1 for the score) are then passed into a classifier for further training as explained and explored in Sections 4.2.2 and 4.2.3. The trained classifier is then saved and ready for the testing stage.

In the testing stage, similar steps are followed, but rather than taking an image from training; it is taken from the validation datasets, where the trained classifier produces the new scores (and labels) as the final outputs.

4.2.2. Encoding Classifier Inputted Features

As illustrated in Figure 4.2, for each scene (image), a detector is applied to produce the bounding boxes (*i.e.*, [x,y,w,h]), objects labels and confidences. Those outputs are then used to find relationships between objects. For each object detected, the outputs from the detector are used to compute the contextual relationships among the detected object with others (only images with more than one object detected are used in experiments).

$$\mathrm{L_{FeatureVector}} = \mathrm{Conf_{Ref}} + (\mathrm{L_{Relation}} \times 80) \tag{4.1}$$

The desired relationship(s) is/are calculated following the mathematical equations presented in Tables 3.1 and 3.2, in Chapter 3. The dimension of the encoded features varies upon the number of relationships used. Equation 4.1 shows how the dimension of the feature vector is computed, where $L_{FeatureVector}$ represents the length of the feature vector inputted into the classifiers, $Conf_{Ref}$ is the confidence value of the reference object, and $L_{Relation}$ is the length of relationships as presented in Table 4.1. 80 is the number of the object classes as in MSCOCO 2017 dataset, which are listed in Table 2.1, Chapter 2.

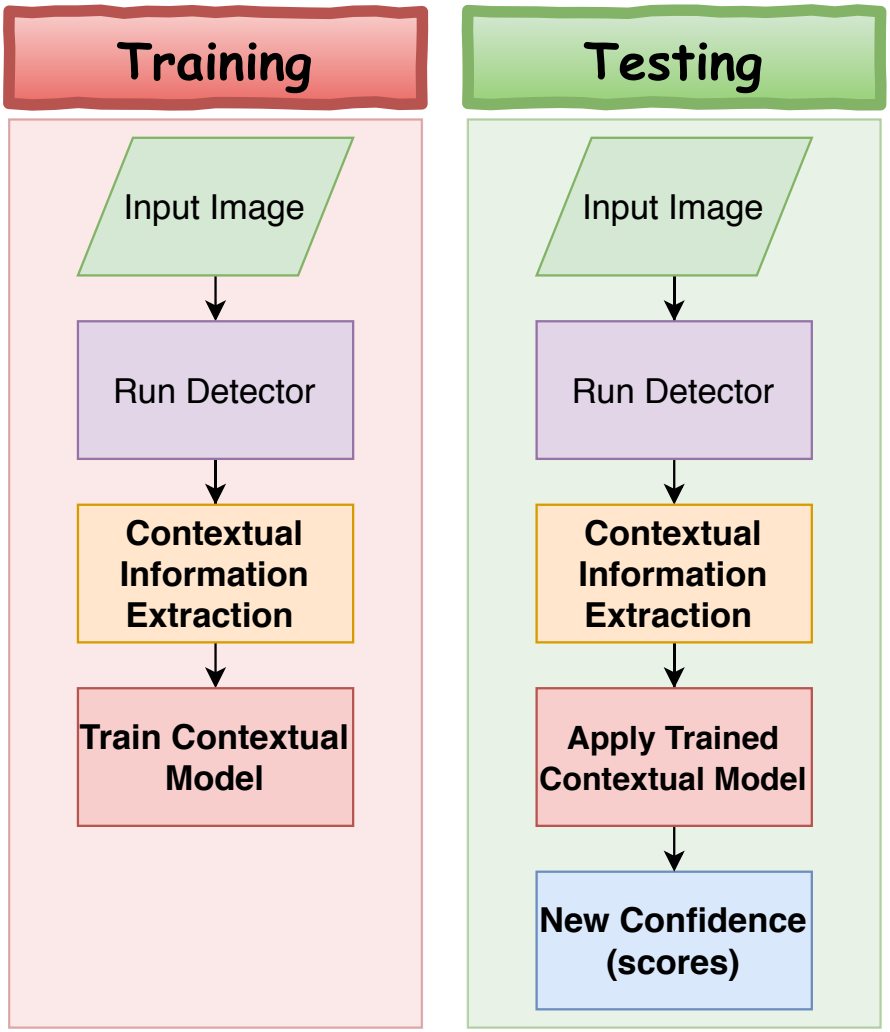


Figure 4.1.: Procedure of the Proposed Rescoring Model

For example, the length of feature vector in terms of the co-occurrence relationship is 1 + (1 × 80) = 81. As shown in Figure 4.3, an input image (taken from MSCOCO 2017) is shown on the left, and detector output (*i.e.*, Faster RCNN) is shown on the right side. Only two objects were detected, which are a `person` and a `skis`, with confidence scores of *0.9998* and *0.9427* respectively. As the class `person` is ordered as the 1st in the MSCOCO dataset classes, and class `skis` is the 31st. Therefore, only the positions 1st, 2nd and 32nd in the features are filled as shown in Figure 4.4. Position 1st includes the confidence of the reference object (*i.e.*, `person` in the second row and `skis` in the third, where the first is the dimension of the feature). Positions 2nd and 32nd include the scores for classes person and skis, respectively. As it can be seen in the second row, the confidence of the `person` is filled in the first column, and all others are zeros apart of the 32nd column, which includes the score for the `skis`, as it is the only other object detected in the inputted image. In case there are two instances of the same class detected (*i.e.*, two `persons`), only the instance with the maximum confidence is considered. Once features are ready and encoded with the desired relationship(s), they are inputted into the classifier, as explained in Section 4.2.3.

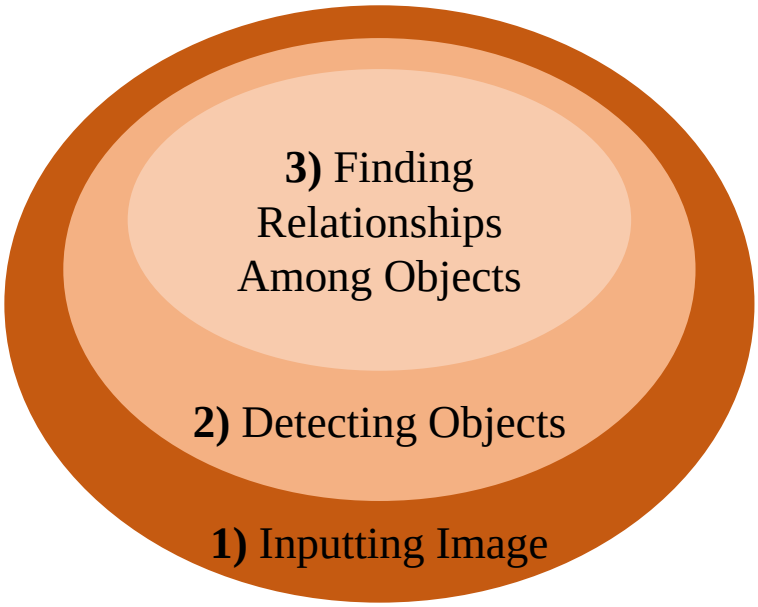


Figure 4.2.: Encoding Relationships into a classifier

(a) Original Image (b) Detector Output

Figure 4.3.: Model Input Feature Vector

Table 4.1.: Length of feature vector per relation.

Relationship	Number of features per relations	Length of the feature vector
Co-occurrence	1 (either co-occur or not)	81
Overlapping	2 (Yes, No)	161
Scale	3 (Large, Small, Equal)	241
Boundary	4 (above, Below, Left, Right)	321
Central	4 (above, Below, Left, Right)	321
Near Far	2 (Near, Far)	161
All Relations	**Sum of all above**	**1281**

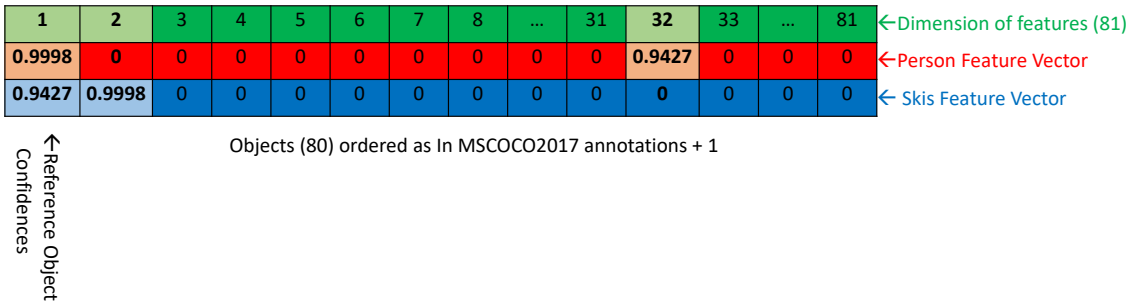


Figure 4.4.: Example of the Classifier Input Features

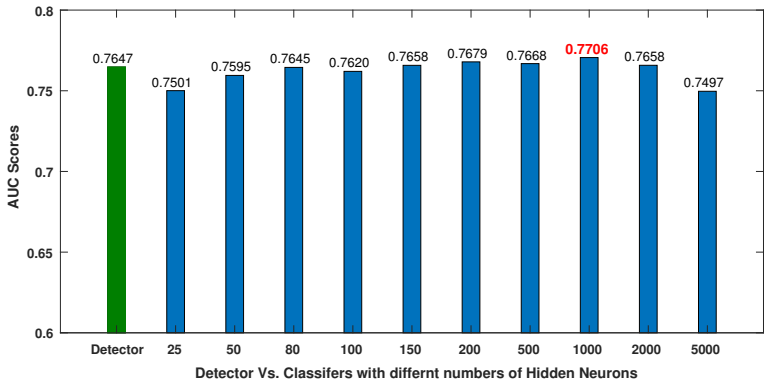


Figure 4.5.: Implementation of the classifier with different numbers of hidden neurons: Red bar represents the detector score, whereas the blue bars represent the scores of the classifiers with different number of hidden neurons

4.2.3. Classifier

For the experiments in this work, I use a `trainscg` (scaled conjugate gradient backpropagation) Neural Network function, as proposed in MATLAB (MATLAB, 2017). Scaled conjugate gradient (SCG), a supervised learning algorithm, is a network training function used to update weight and bias value according to the scaled conjugate gradient method (Møller, 1993). This applied classifier is `trainscg`, which is implemented as explained here MATLAB (2017). SCG is chosen as suggested by MATLAB in regards to accuracy and speed in case of large data, as the case of this work. A shallow neural network, which consists of a two-layer feed-forward network, with a sigmoid transfer function in the hidden layer, and a softmax transfer function in the output layer is used. Several numbers of hidden neurons were tested (*i.e.*, 25, 50, 80, 100, 150, 200, 500, 1000, 2000, 5000), and classifiers with 1000 Hidden Neurons (HNs) performed as the best on MSCOCO 2017 (even better than Faster RCNN), as AUC scores shown in Figure 4.5. Therefore, the Neural Network with 1000 HNs is chosen and used in all experiments presented in this chapter to rescore detected objects confidences (and relabel them).

Other classifiers were also applied and evaluated, which are K-Nearest Neighbours (KNN), Decision Tree, and Bag of Decision Tree (BoDT) (See Appendix B.1, for more information). Due to the performances and speed of those classifiers (in both training and testing) compared to NN classifiers, they are not applied. In all the experiments (Section 4.2.4 in this Chapter, NN classifier with 1000 HNs is used.

Table 4.2.: AUC Scores and STD: One-Relationship Model

One Contextual Relationship Model	AUC Scores (STD)
Co-occurrence	**0.76658** (0.0015)
Boundary	0.75855 (0.0016)
Central	0.75888 (0.0011)
Overlapping	**0.77319** (0.0017)
Near/Far	**0.76634** (0.0010)
Scale	**0.76696** (0.0012)
Baseline Detector (Faster RCNN)	0.76472

4.2.4. Experiments

In this section, several experiments have been attempted to examine the impact of the proposed *Rescoring Model* upon the use of two different approaches baseline detectors (Faster RCNN and YOLO) and two different datasets (MSCOCO 2017 and PASCAL 2012). Experiments, as stated below, are to present the impact of each contextual relationship individually, a combination of two, three and four relationships and then all relationships combined.

Experiment One: One Contextual Relationship Model

In this experiment, each type of the proposed sixteen contexts is examined individually to investigate their impacts on the performance of the detection, and how they can rescore detected objects' confidences. Understating the impact of each relationship can help to explore deeper comprehension of how objects interact.

Each of the relationships is applied as post-processing upon the use of the baseline detector (Faster RCNN) and on MSCOCO 2017, and then evaluated. Evaluation is made in comparison with the baseline detector, using AUC scores, as presented in Table 4.2. AUC scores are the only evaluation method used in this experiment due to its efficiency, as only the objects confidences are measured (object labels are not corrected here).

As presented in Table 4.2, it can be seen that most of the relations obtain higher AUC scores than the Faster RCNN. Only the average AUC scores for all 80 classes in MSCOCO 2017 is reported, where all scores are presented in Appendix B.2. Standard Deviation (STD) values for each relationship is also presented as shown between brackets to show the difference in scores where five trials are used for each relationship (See Appendix B.3, for all scores obtained per class per each trail, as for each relationship).

In detail, overlapping, as seen, has the highest AUC scores among all other models, including the detector. This could to be because objects are either overlapping or not, so that it is reducing the variations in data, where more objects may have the same overlapping relations even in different images. For example, it is more likely that an airplane would not be overlapped with a knife, because they are unlikely to be seen in one image, which, in other words, encodes semantic relationship implicitly. Scale, co-occurrence and distance

relationships have almost similar scores, which are still higher than Faster RCNN. However, boundary and central relationships obtain lower than the baseline detector, which could be due to the complexity of those relations and the huge variations in data, as each consists of four sub-group relationships (*i.e.*, above, below, left and right). In general, based on the scores reported in Table 4.2, it can be noticed that contextual information does improve detection performance, which is in line with most of the studies concerning this issue.

Visual results are also important to illustrate the impact of context, as reported in Table 4.4. It is shown some results obtained during running one-relation models. Images for this experiment are selected from MSCOCO 2017 validation dataset. For simplicity and better visualisation, only the outputs from the detector and one-relationship model are shown on the left side and right side, respectively.

In the first row, semantic relation model is applied, and it can be seen how the scores of all three detected objects increased, that could be due to the high co-occurrence between such objects, as they are more likely to be presented in the same scene since all belong to one main environment (*i.e.*, living room). As presented in Table 4.3, the top ten objects co-occurred with the detected objects excluding the detected object itself are reported, (See Appendix A, for the full co-occurrence matrix). It can be clearly seen that `chair`, `couch` and `TV` are one of the top ten objects co-occurring with each other, where all other objects in the top ten list are likely to present in the same location. Therefore, this can be the motive behind the co-occurrence-relationship model for increasing those objects confidence. The `chair`, for example, increased from *0.8953* to *0.9808*.

Second and third rows of images, where the boundary and central relationship models are applied, it can be seen how boundary relationship model performs poorly by reducing the objects' confidences, but it could be due to the low number of images where both detected objects (*i.e.*, `suitcase` and `bed`) appear in the same scene. Only 204 images contain both objects, where number of images containing `suitcase` is 2402, and `bed` is 3682 (as in the matrix presented in Appendix A). Therefore, it can be said that if there were more images provided in training, the model might have performed better. However, in the central relationship model, we can see that only `persons` confidences are increased, where other objects scores are decreased, as `clock` is incorrectly detected. This could be due to the large variations in relationship applied, as stated earlier.

In terms of the overlapping relationship model, it has the highest AUC score among all other one-relationship models and the baseline detector. As the result shown in the fourth row in Table 4.4, we can see that the `cow` is incorrectly detected as a `house`; hence the overlapping model decreases its confidence. This could be an example shown how effective is the contextual information, answering Q1 (Section 1.2) that this work intends to address, which is "*To what extent do semantic, spatial and scale relationships enhance the detection performance?*, where the answer can be clearly seen.

For the results presented in the following row, near/far relationship model is applied. Several objects are incorrectly detected, where the model even reduces their confidences. However, for the correct-detected objects, it can be said that due to the low number of

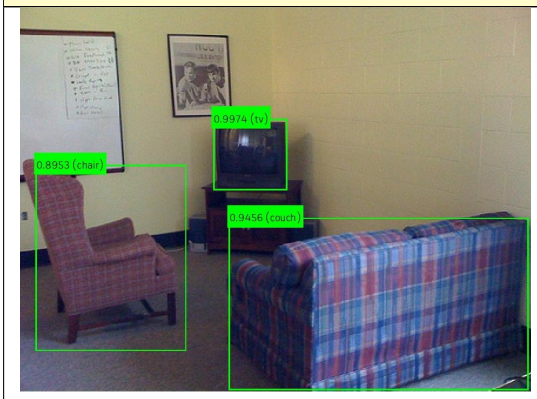
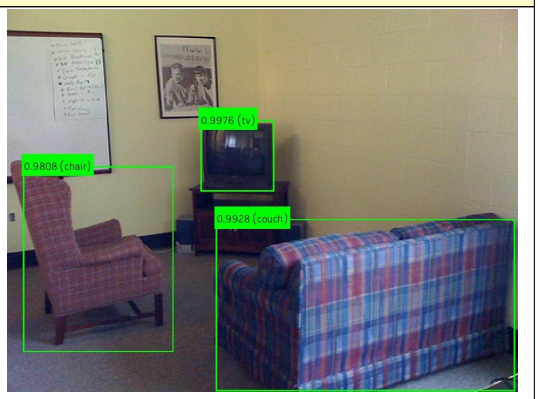
Table 4.3.: Top ten objects co-occurring with `Chair`, `Couch` and `TV`: objects names are presented, where semantic statistics are also shown between the brackets. Such statistics are obtained from the MSCOCO 2017 training dataset.

Chair	Couch	TV
Person (0.6290)	Person (0.5055)	Chair **(0.4363)**
Dining table (0.4011)	Chair **(0.4851)**	Person (0.4249)
Cup (0.2307)	Remote (0.3271)	Book (0.3225)
Bottle (0.1863)	Book (0.3081)	Couch **(0.2786)**
Couch **(0.1679)**	TV **(0.2873)**	Keyboard (0.2745)
Book (0.1651)	Dining table (0.2333)	Remote (0.2468)
TV **(0.1557)**	Potted plant (0.2109)	Mouse (0.2337)
Bowl (0.1396)	Vase (0.1765)	Laptop (0.2227)
Potted plant (0.1345)	Cup (0.1627)	Cup (0.1878)
Laptop (0.1070)	Bottle (0.1329)	Dining table (0.1655)

correctly detected objects, the confidence of this object is reduced. For example, detecting the `cup`, which is unlikely to be seen in such location with respect to the `toilet` class.

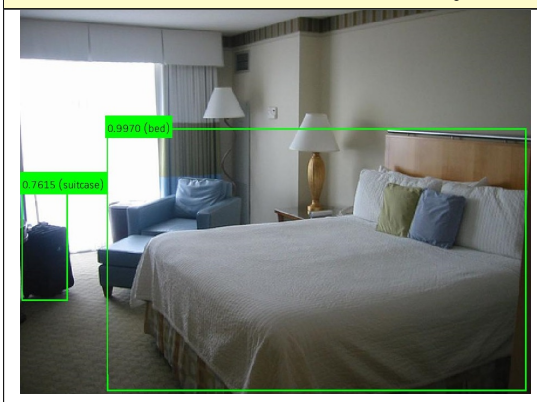
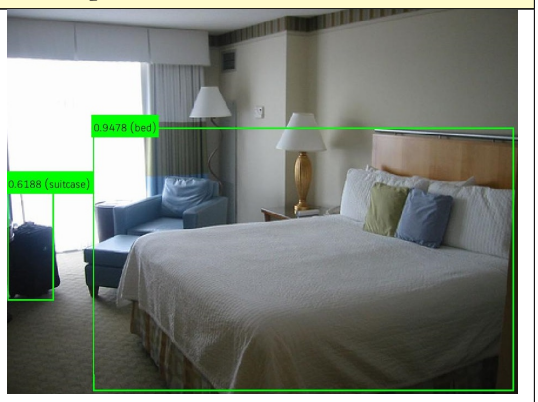
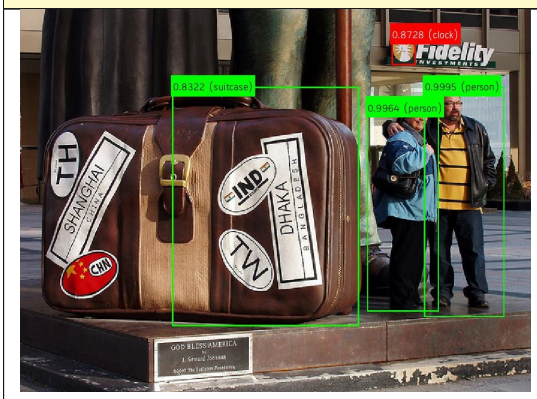
Finally, scale-relationship model as illustrated in the last row. We can see that `tie` confidence is increased, as its smaller in size compared to `person`. Such objects are expected to be well-training as more than 90% of the images where `tie` and `person` are presented, it is found that 91% `tie` is smaller, 8.9565% `tie` is larger, 0.0037% `tie` is equal in size with `person` (See Appendix A, for more statistics).

Table 4.4.: One-Relationship Model vs. Faster RCNN Detector outputs: green boxes represent correct detection, whereas red are incorrect (or not in the ground-truth)

Faster RCNN Detector	**One-Relationship Model**
Co-occurrence Relationship Model	
	

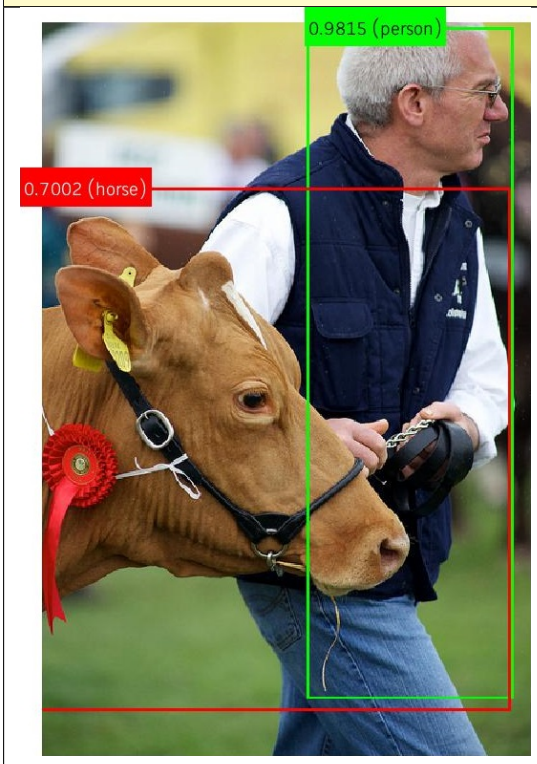
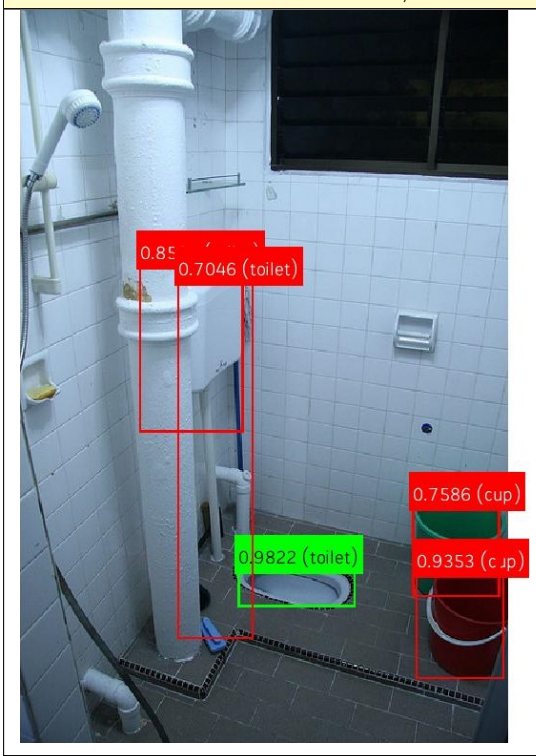
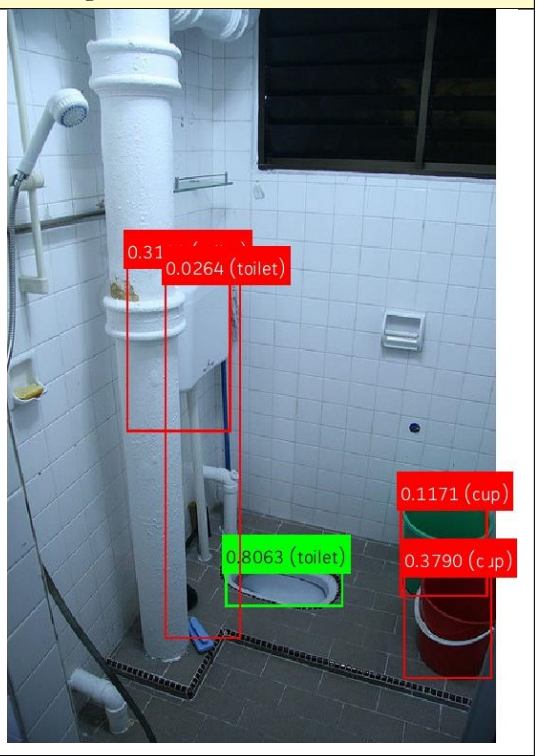
Continued on next page

Table 4.4 – continued from previous page

Faster RCNN Detector	One-Relationship Model
Boundary Relationship Model	
	
Central Relationship Model	
	

Continued on next page

Table 4.4 – continued from previous page

Faster RCNN Detector	One-Relationship Model
Overlapping Relationship Model	
	
Near/Far Relationship Model	
	

Continued on next page

Table 4.4 – continued from previous page

Faster RCNN Detector	One-Relationship Model
Scale Relationship Model	
	

In short, examining the use of the five categories of the proposed contextual information results to the conclusion that most of them do help to improve the detection performance. Therefore, combinations of two, three, and four relationships are applied to evaluate how such combinations can lead to an increase in the performance, as explored below.

Experiment Two: A combination of Contextual Relationships Model

In this experiment, a combination of relationships is applied. This experiment is demonstrated to examine how a combination of relationships models can impact on the performance of the baseline detector, where Faster RCNN is used. A combination of two, three and four relationships are presented as follows. The idea behind such a combination comes from the assumption that the more we understand the context around objects, the more it is expected to improve the detection.

- A Combination of Two-Relationships:

In this sub-experiment, only two relationships are combined, as shown in Table 4.5. As presented, it can be seen that most of the two-relation models are performing with higher AUC scores compared with the baseline detector (AUC score is *0.76472*, as presented in Table 4.2), where the highest among those models is co-occurrence and overlapping combination scoring *0.77296*. Such a result is expected to some extent, as overlapping itself performs as the highest as reported earlier in Section 4.2.4. However, we can also see that co-occurrence-boundary, co-occurrence-central, co-occurrence-scale and central-Near/far models have lower scores than Faster RCNN.

Table 4.6 presents some results obtaining from applying some of the two-relationships models, detector outputs are in the left column, whereas two-relationships models are on the right. In the first row, Boundary-Central relationships model is applied. As noticed, the detector detects three objects, which are two `airplanes` and a `bus` with confidences *0.9953*, *0.9916*, and *0.7515* respectively. The contextual model then reduces confidences to *0.9652*, *0.9468* and *0.4324* as for the two `airplanes` and the `bus` respectively. Yes,

Table 4.5.: AUC Scores and STD: Two-Relationship Models.

Relationships	Boundary	Central
Co-occurrence	0.76364 (0.0010)	0.76413 (0.0015)
Boundary	-	0.75647 (0.0005)
Central	0.75647 (0.0005))	-

Relationships	Overlapping	Near/Far	Scale
Co-occurrence	**0.77296** (0.0012)	**0.76736** (0.0017)	0.75811 (0.0017)
Boundary	**0.77162** (0.0011)	**0.76760** (0.0018)	**0.76895** (0.0013)
Central	**0.76775** (0.0010)	0.75296 (0.0010)	**0.76648** (0.0017)

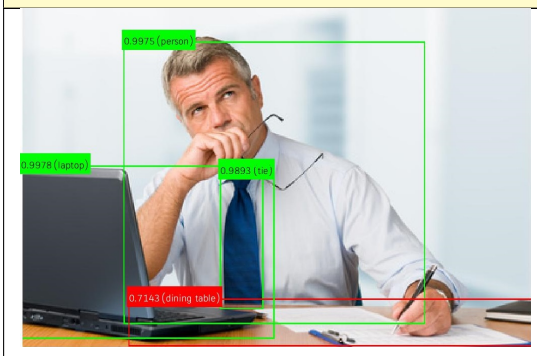
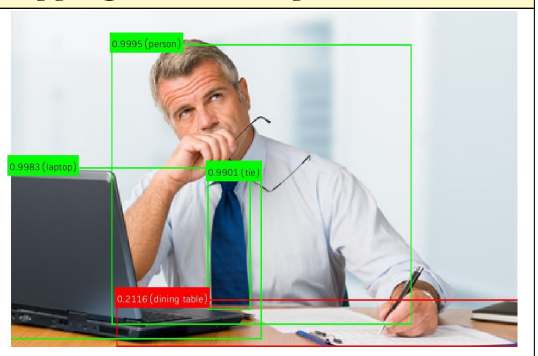
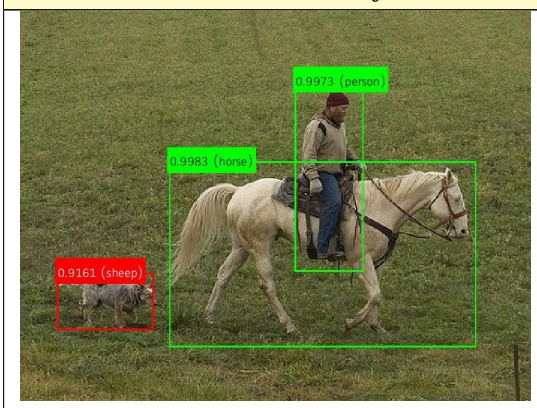
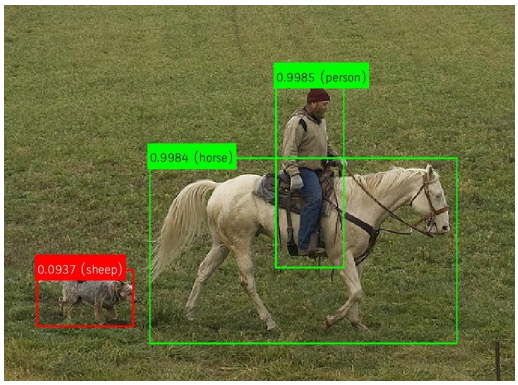
there is not a significant change observed in scores in terms of `airplanes`, which seems to be expected as this model has lower AUC scores compared with the baseline detector. However, it can be clearly noticed, even when the performance of such a model is low, it helps to reduce the `bus` confidence, which can be considered as a good move, as such a class is incorrectly detected.

Moreover, Co-occurrence-Overlapping model, which performs as the best among all other models including the baseline detector (*i.e.*, it has the highest AUC scores), is applied as the results presented in the second row. It can be seen how all objects confidences have improved and increased due to the application of such a model. Three objects, which are a `person`, a `tie`, and a `laptop`, are correctly detected, whereas a `dining table` is incorrectly detected. The context model is then applied, which changes confidences for the correct-detected objects from *0.9975*, *0.9893* and *0.9978* to *0.9995*, *0.9901* and *0.9983* respectively. This is due to the overlapping and semantic relationships between those objects, as they are, in this image, presented in positions, where they likely to be seen in. However, in terms of the incorrect-detected objects, the model considerably decreases its scores from *0.7143* to *0.2116*. This clearly illustrates how solid the model is and how it is expected to improve objects scores upon the use of contextual constraints.

In the third row, a visual result outputted from applying the Boundary-Overlapping model is shown. This model, as reported earlier in Table 4.5, has the second highest AUC score. The model significantly drops the scores of the `sheep` (*i.e.*, incorrectly detected, as its actually a `dog`), from *0.9161* to *0.0937*. However, it slightly enhanced the scores for the correct-detected objects (*i.e.*, `person` and `horse`), as from *0.9973* to *0.9985* for `person`, and from *0.9983* to *0.9984* for the `horse`. This, therefore, shows another example of how applying such context models preform well.

On the other hand, Central-Overlapping model performs poorly as presented in the last row. As illustrated, even though the model has a higher AUC score compared to the detector, it reduces the scores of all objects while they are correctly detected. This can be because of the complexity of the scene, and how objects are interrelated and located.

Table 4.6.: Two-Relationships Model Vs. Faster RCNN Detector outputs: green boxes represent correct detection, whereas red are incorrect (or not in the ground-truth).

Faster RCNN Detector	Two-Relationships Model
Boundary and Central Relationships	
	
Co-occurrence and Overlapping Relationships	
	
Boundary and Overlapping Relationships	
	

Continued on next page

Table 4.6 – continued from previous page

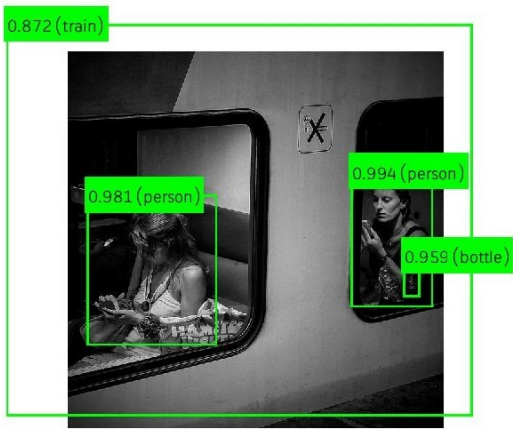
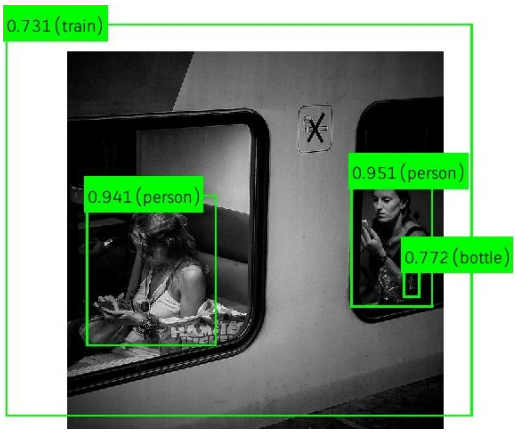
Faster RCNN Detector	Two-Relationships Model
Central and Overlapping Relationships	
	

Table 4.7.: AUC Scores and STD: Three-Relationship Models.

Relationships	AUC Scores (STD)
Co-occurrence + Boundary + Central	0.75930 (0.0019)
Co-occurrence + Boundary + Scale	**0.76812** (0.0018)
Co-occurrence + Central + Scale	**0.76513** (0.0015)
Boundary + Central + Scale	**0.76825**(0.0021)

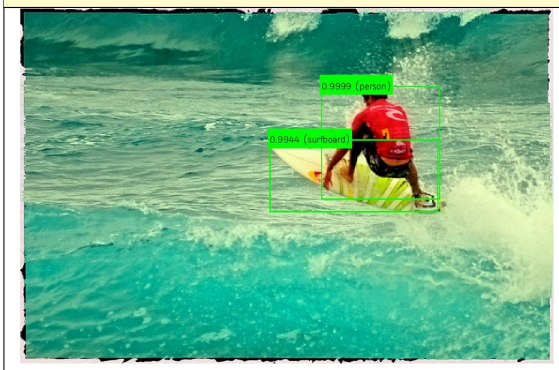
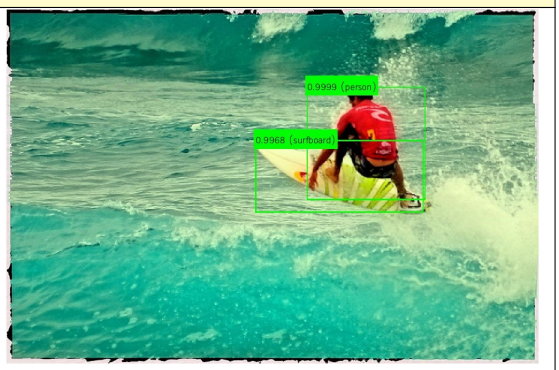
- A Combination of Three-Relationships:

In the section, three-relationship model is applied. The three-relationships examined in this experiment are randomly chosen, as not every combination of three-relationships is attempted. As shown in Table 4.7, all the reported three-relationships models obtain higher AUC scores than Faster RCNN, except the combination of co-occurrence, boundary and central relationships. This can be due to the low performance of boundary and central relationships, which may lead to drawing the conclusion that boundary and central relationships due to the high variations as discussed and reported in Section 4.2.4.

Refer to Table 4.8 for some visual results illustrating some of those three-relationships models. Two three-relationships models are applied, which are co-occurrence, boundary and central model and co-occurrence, central and scale model. The former is applied in the first row: the detector first detects a `person` and a `surfboard`. The model is then applied, which slightly increases the scores from *0.99988* to *0.99989* for `person`, and from *0.9944* to *0.9968* for the `surfboard`. This can be considered, to some extent, as good results for this model, because it scores lower AUC scores compared to the baseline detector as presented in Table 4.7. In the second row, the latter examined model is applied (*i.e.*, co-occurrence, central and scale model). We can notice that correct detected objects scores are increased, whereas scores for the incorrectly detected objects are reduced by the model. This is exactly what we expect from such models to perform. In detail, the confidence of the incorrectly detected `car` (*i.e.*, the red squared on the right shoulder of `person` on the left)

drops from *0.7031* to *0.1976*. This is because the size and spatial relationships between this object and other detected objects are unlikely to occur. Similarly, the confidence for `person`, which is incorrectly detected as well, as the actual object is a `towel`, is lowered from *0.8015* to *0.2622*.

Table 4.8.: Three-Relationships model vs. Faster RCNN Detector outputs: green boxes represent correct detection, whereas red are incorrect (or not in the ground-truth)

Faster RCNN Detector	Three-Relationships Model
Co-occurrence, Boundary and Central	
	
Co-occurrence, Central and Scale	
	

- A Combination of Four-Relationships:

Finally, a combination of four-relationships is tested, as presented in Table 4.9. Only two combinations of four relationships are reported, which have been randomly chosen. It can be seen how the models in the two cases perform better compared with Faster RCNN. In comparison between the two models, the one, which has central relationships included, perform lowers than the other, where all other three relationships are the same.

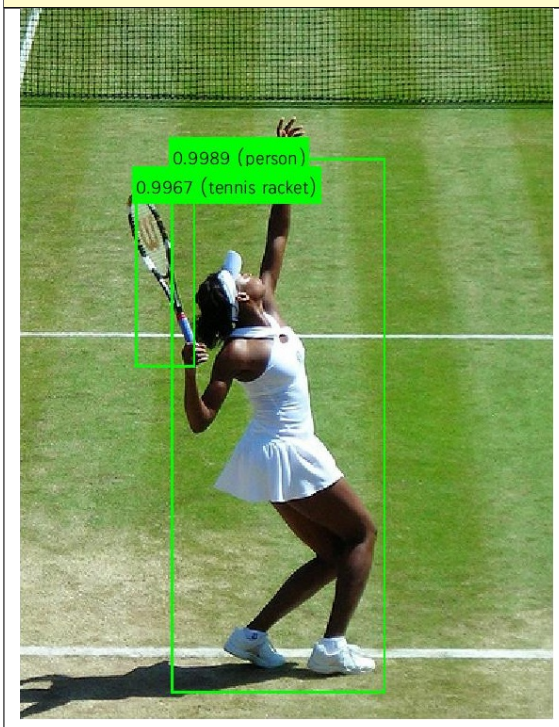
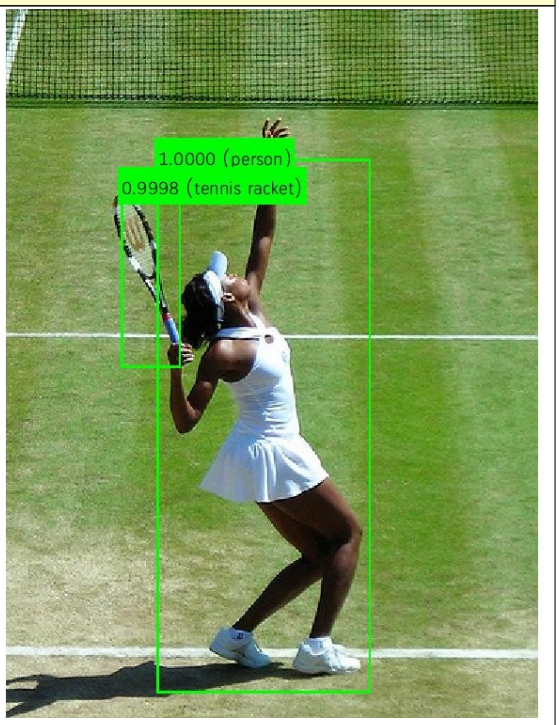
Table 4.9.: AUC Scores and STD: Four-Relationships Models.

Relationships	AUC Scores (STD)
Co-occurrence + Boundary + Scale + Overlapping	**0.77172** (0.0007)
Co-occurrence + Central + Scale + Overlapping	**0.76733** (0.0017)

Table 4.10 shows the results of applying co-occurrence boundary overlapping and scale relationships model (*i.e.*, one of the four-relationships models applied). It can be noticeable

that the model increases the scores of both correct-detected objects, which are a `person` and a `tennis racket`, from *0.9989* and *0.9967* to *1* and *0.9998*, respectively. This is, predictable to occur due to the high performance of this model, and the likelihood of such objects to occur in such scales and positions. Looking at matrices presented in Appendix A, it can been seen that 99% of cases where a `tennis racket` and a `person` are presented: 73% of them `tennis racket` is overlapping with `person`, and 72% of the cases `person` is larger in size compared to `tennis racket`. Upon such statistics, it can be expected that the model to perform as it did, as this case is similar to the majority of cases where the classifier is trained on.

Table 4.10.: Four-Relationships model vs. Faster RCNN Detector outputs: green boxes represent correct detection

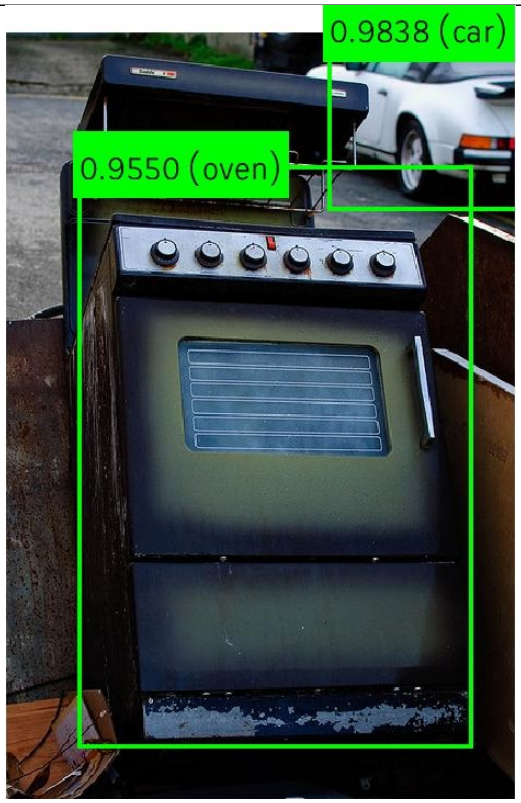
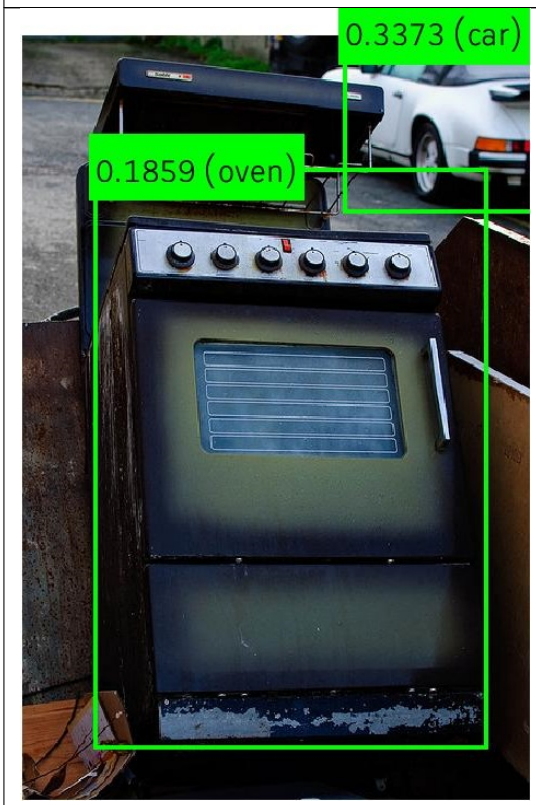
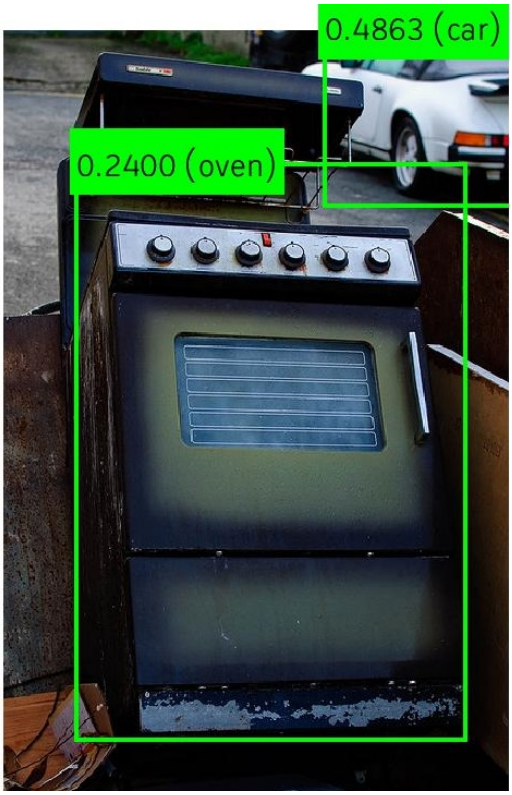
Faster RCNN Detector	**Four-Relationships Model**
Co-occurrence, Boundary, Overlapping and Scale	
	

- A Comparison between one-relationship models and a combination of relationships models:

In this section, different combinations of relationships are examined on the same input images. This is to illustrate how each relationship (and a combination of relationships) can re-rate the prediction scores upon the contextual information extracted from the scene. Two different input images are used for this experiment.

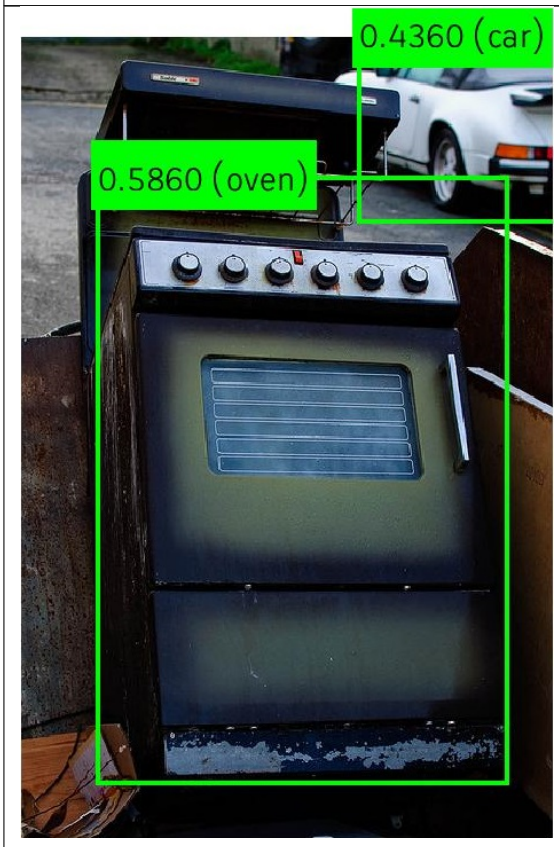
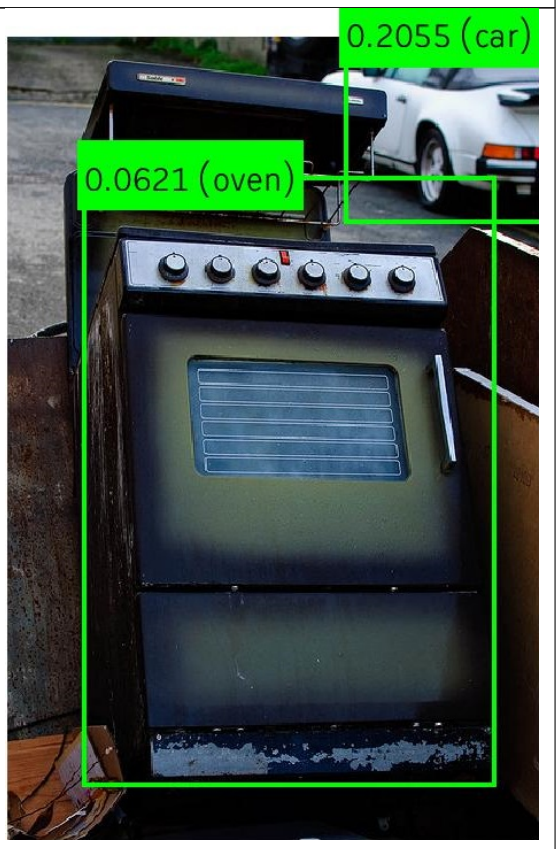
As presented in Table 4.11, the first inputted image is shown, which consists of a car and an oven. It is then passed into Faster RCNN, which detects both objects correctly with very high confidence scores (*0.9838* for the `car`, and *0.9550* for the `oven`). Overlapping relationship model is then applied, which considerably reduces both scores to (*0.3373* for the `car`, and *0.1859* for the `oven`). Even though both objects are detected correctly, but they luck contextual consistency. The `car` is likely to be seen as an outdoor object, whereas the `oven` is an indoor object. As shown in the overlapping matrix in Appendix A that there is no overlapping relationship between both objects. This is assumed to be the reason for this scores reduction. Similarly, as shown in the second row, Co-occurrence and Overlapping relationships model is applied, which also reduces the scores, but still higher than when Overlapping relationship model is applied. After looking at the Co-occurrence matrix for the `car` and `oven`, it was found that there is a very slight number of images where both appear in the same image (but not overlapped). Three relationship model, which is Boundary, Central and Scale relationships model is applied. This shows a different result compared to the models already applied to this image. It reduces both scores, but assign the `oven` with higher score compared to the `car`. This can be due to models' low performance (as the AUC scores show in Table 4.7). Co-occurrence, Boundary, Scale and Overlapping relationships model is applied as shown on the right side of the last row in Table 4.11. This model also reduces all objects scores (*i.e.*, *0.2055* for the `car`, and *0.0621* for the `oven`). All model are seen to reduce the scores even when objects are correctly detected, but looking at their contextual consistency, they are seen to unlikely to appear in the same image, due to the difference in context they are likely to be presented in. This image is chosen for this illustration, to present the impact of context, and how it depend upon contextual features among objects. Even though objects are correctly detected, but they are unlikely to co-occur (upon the contextual statistics extracted from MSCOCO 2017 training dataset).

Table 4.11.: A visual comparison between single-relationship models and multiple-relationship models: green boxes represent correct detection, whereas red are incorrect (or not in the ground-truth)

Input Image	Faster RCNN Output
	
Overlapping	**Overlapping and Co-occurrence**
	

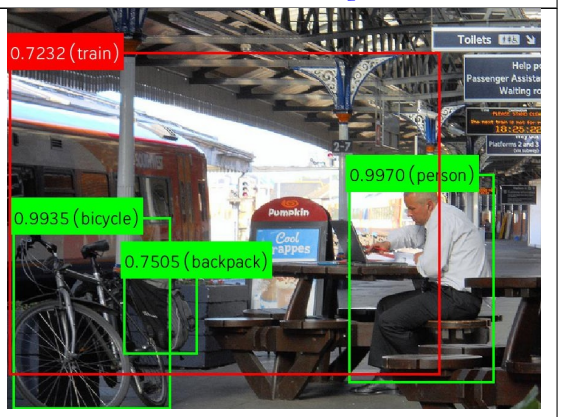
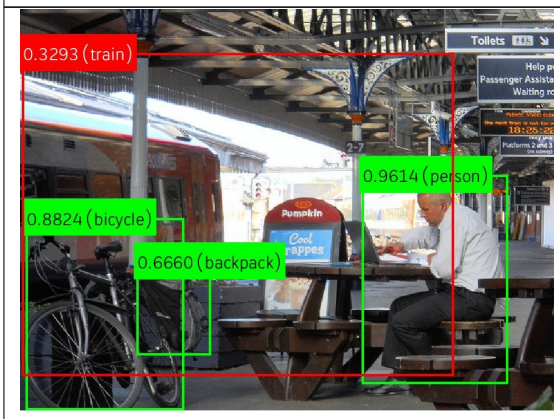
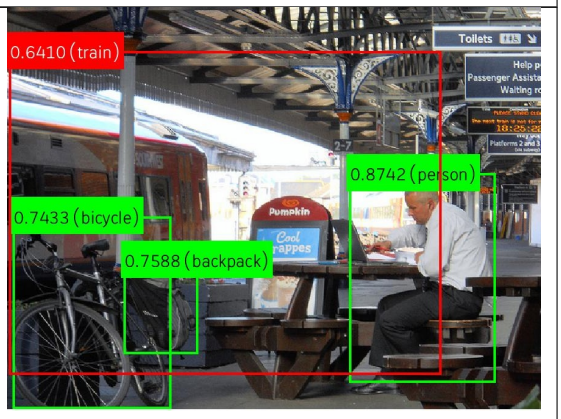
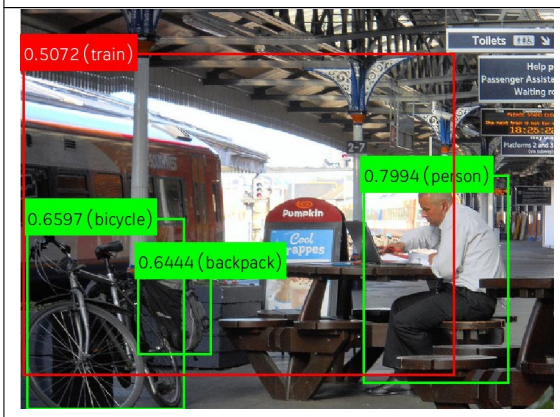
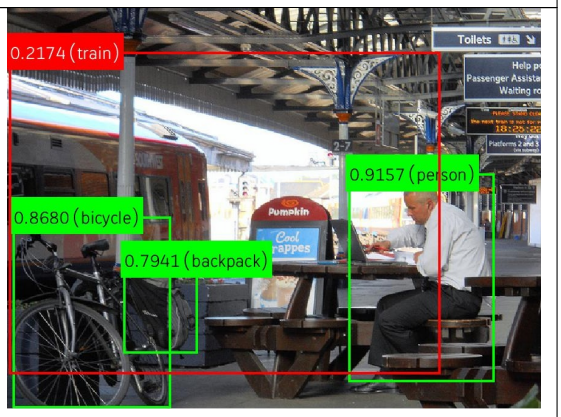
Continued on next page

Table 4.11 – continued from previous page

Boundary, Central and Scale	Co-occurrence, Boundary, Scale and Overlapping
	

Moreover, Table 4.12 shows other results obtained when examining a different set of relationship on a different input image. The input image is shown on the right side of the first row, next to the detector outputs, which are shown on the right side. The detector detects a `person`, a `bike` and a `backpack` correctly, but incorrectly detects a `train`. Co-occurrence relationship model is then applied, which reduces all scores, particularly the `train` score is significantly dropped to *0.3293*, as it is incorrectly detected. Followed by applying a two-relationship model, which is Co-occurrence and Scale relationships model. This model also reduces all scores, but keeping all correct detections as higher than *0.74*. Co-occurrence, Boundary and Scale relationships model, as three-relationship model is applied. This seems to perform similarly as the two-relationship model applied as it reduces all scores, where the incorrect detected object score is also dropped to *0.5072*. Finally, as shown in the last row Table 4.12, four-relationship model, which is Co-occurrence, Central, Scale and Overlapping relationships model is applied, which performs as the best among all models applied to this image. It reduces the `train` score to *0.2174*. This can be due to the inclusion of the overlapping relationship, as it performs as the best among all other relationships. Moreover, as presented in Tables 4.2, 4.5, 4.7 and 4.9 that this four relationship model archives the highest AUC scores among all other applied models to this image.

Table 4.12.: A visual comparison between single-relationship models and multiple-relationship models: green boxes represent correct detection, whereas red are incorrect (or not in the ground-truth)

Input Image	Faster RCNN Output
	
Co-occurrence	**Overlapping and Scale**
	
Co-occurrence, Boundary and Scale	**Co-occurrence, Central, Scale and Overlapping**
	

It can be said that there is no one optimal model that can be applied to all images to achieves the best results, this is because it is dependent on what object classes are presented and how objects are contextually related to each other. For example, having a scene consists of a `person` and a `dog`, they are likely to appear together, which then may help co-occurrence model to perform very well. However, if the `dog` was shown as larger than

a `person`, which is unlikely to occur, as the statistics show in Appendix A: a `person` is likely to be larger in size than a `dog`. Therefore, this can lead the scale model to poorly perform. Combining all relationships into one model, as presented in Experiment 4.2.4, is seen to overcome such a limitation. Therefore, due to the great improvements obtained in the performance upon the use of the contextual relationships (as an individual or combinations), combining all proposed relationships into one model is performed, named as the *Rescoring Model* as presented below.

Experiment Three: All Contextual Relationships Model

In this experiment, due to the great success obtain during examining each relationship and several combinations of relationships individually as presented in Experiments One and Two above, combining all the relationships into one model seems as a need to be examined. Therefore, all sixteen contextual relationships are combined into one model, named as the "***Rescoring Model***". As this model is set as to combine all relationships, and to be the *Rescoring Model* proposed, unlike the previous experiment, where only one threshold value for Faster RCNN is used (*i.e.*, 0.7), several detection threshold values, which are [0.5, 0.6, 0.7] are applied in this experiment. This is because it is assumed that lowering the value of the threshold may enable the detector to detect more objects (regardless whether they are correctly detected or not). Therefore, more objects can be detected, which may enable the proposed contextual *Rescoring Model* to understand the context of the image inputted to produce solid outputs. The *Rescoring Model* is tested in comparison with Faster RCNN, where, in all the cases the threshold values are set differently, the model is performing better, as shown in Table 4.13. It can be seen that the *Rescoring Model* in all the case outperforms Faster RCNN as it re-rates objects confidences based on their vicinity. It is almost 0.01% increment obtained upon the use of *Rescoring Model* in all three cases. In the case where the threshold is set as 0.7, as in previous experiments, the average AUC scores for the *Rescoring Model* is *0.771*.

Table 4.13.: AUC Scores: Rescoring Model vs. Faster RCNN.

Threshold Value	Faster RCNN	Rescoring Model
0.7	0.76472	**0.77057**
0.6	0.77911	**0.78562**
0.5	0.79303	**0.80423**

For visual results, *Rescoring Model* is run on some selected images, taken from MSCOCO 2017 validation dataset, as illustrated in Table 4.14. As stated earlier, the aim of the *Rescoring Model* in to re-rate detected objects confidences upon their contextual relationships with each other. Therefore, in the first row of Table 4.14, we can see that the model increases all the detected objects confidences, as they are likely to appear in one scene and in such locations and scales. Two `chairs` and a `refrigerator` are detected, with confidence scores of *0.9938*, *0.9431*, *0.9977*, later the model applied, which increases their scores to *0.9962*, *0.9987* and *0.9981* respectively. Clearly, as all detected objects are likely to co-occur, the model, therefore, increases all scores.

In the second row, two objects are correctly detected, which are a `person` and an `over` with confidence scores *0.9868* and *0.8155*, respectively. One object, on the other hand, is incorrectly detected, which is a `keyboard` scored as *0.8899*, which is even higher than one of the correctly-detected objects confidence. The *Rescoring Model* is then applied to re-rate scores, which decreases the `keyboard` score to *0.3681*, which is believed to be due to its low co-occurrence other objects. Particularly, looking at the the `oven` and the `keyboard` co-occurrence statistics, it is found that only 11 images (out of 2115, as the number of images containing keyboard), which contain the `keyboard` and the `oven` together. However, even `person` and `oven` may appear together, but their scores change, as `person` score is reduced *0.9715*, whereas `oven` score increased to *0.8491*.

Moreover, in the third row, we can see that the model does not perform as expected, where it reduces all detected objects confidences. For example, the `TV` score is reduced from *0.8686* to *0.1972*. I assume this scene is even hard for the detector, to some extent, as none of the objects detected has confidence higher than 0.9 except the `sink` which is even incorrectly detected, as it is actually a `bath`. Similarly, in the following fourth row, the model performs slightly poorer than expected. It decreases all detected objects scores. Due to the presence of three `remotes`, it is expected that the model would, at least, increase their confidences. However, when the co-occurrence matrix is looked at, it is found that the semantic relationship between `remote` and `bed` is low, as `remote` is more likely to be seen with `chairs`, `TVs` and `couches`. However, looking at the majority of visual results and the statistical analysis, I can say that this model performs as an excellent tool re-rating objects scores upon their contexts, which is even scores a higher average AUC score than Faster RCNN.

Table 4.14.: Rescoring Model vs. Faster RCNN Detector outputs: green boxes represent correct detection, whereas red are incorrect (or not in the ground-truth)

Faster RCNN Detector	Rescoring Model
	
	
	

Continued on next page

Table 4.14 – continued from previous page

Faster RCNN Detector	Rescoring Model
	

To ensure the efficiency of the *Rescoring Model*, it is examined on PASCAL 2012 dataset, where the baseline detector is Faster RCNN, but only one threshold value is used, which is 0.7. Results confirm that the *Rescoring Model* also performs better than Faster RCNN, as shown in Table 4.15. The reader is referred to Appendix C, for more statistical results. Similarly, when the *Rescoring Model* is applied on Faster RCNN using a previous version of MSCOCO dataset (MSCOCO 2014), it performs better, as presented in Table 4.15. The latest version of MSCOCO is primarily used because it includes more images in the training dataset compared to previous versions (11828 images in MSCOCO 2017, and 82783 in MSCOCO 2014). This helped to understand objects contexts, as the more examples we train the model, the better the model is expected to be.

Table 4.15.: AUC Scores: Rescoring Model vs. Faster RCNN on Different Datasets.

Dataset	Faster RCNN	Rescoring Model
PASCAL 2012	0.78432	**0.79369**
MSCOCO 2014	0.77964	**0.78252**

Table 4.16.: AUC Scores: Rescoring Model vs. YOLOv1 Detector.

Threshold Value	You Only Look Once	Rescoring Model
0.7	0.66977	**0.67894**

Table 4.17.: AUC Scores: Running the Rescoring Model in Iterations.

Number of Iteration	AUC Scores
1^{st}	0.77057
2^{nd}	0.77116
3^{rd}	0.78570
4^{th}	0.79307
5^{th}	0.80482
6^{th}	0.80537
7^{th}	0.80579
8^{th}	0.80620
9^{th}	0.80690
10^{th}	0.80693
11^{th}	0.80695
12^{th}	**0.80702**

In addition, the *Rescoring Model* is also tested on MSCOCO 2017 compared with another baseline detector, YOLOv1, which implicitly includes contextual information into its end-to-end detection pipeline (Redmon et al., 2015). The *Rescoring Model* performs better than this baseline detector as well, using MSCOCO 2017 dataset, as presented in Table 4.16, where only one threshold value (*i.e.*, value is 0.7) is used. The reader is referred to Appendix C, for all AUC scores for all objects in a comparison between the *Rescoring Model* and YOLO.

Experiment Four: Out-Of-Context

As presented earlier in Experiment Three above, the *Rescoring Model* (*i.e.*, all-relationships model) shows good results, and outperforms the performances of the baseline detectors (*i.e.*, Faster RCNN and YOLOv1) performances, as tested on different datasets (*i.e.*, MSCOCO and PASCAL). Therefore, I deeply studied the model trying to examine if the performance would be even higher by iterating the process of rescoring, where this experiment is only run on Faster RCNN as the baseline detector.

The answer is **Yes**, the performance has increased from *0.77057* to *0.80702* (in the 12^{th} iteration), which is a great step in rescoring the detection outputs, as presented in Table 4.17, which shows how the AUC increases from the 1^{st} run to 12^{th} iteration run, and visualised as in Figure 4.6.

In detail, this iterated process is done as follows: First, I apply the *Rescoring Model* to the detector outputs. Scores obtained from the *Rescoring Model* are then fed again to the *Rescoring Model* for 12 times, where the final outputs (*i.e.*, the 12^{th} run outputs) are considered as the iterated model outputs.

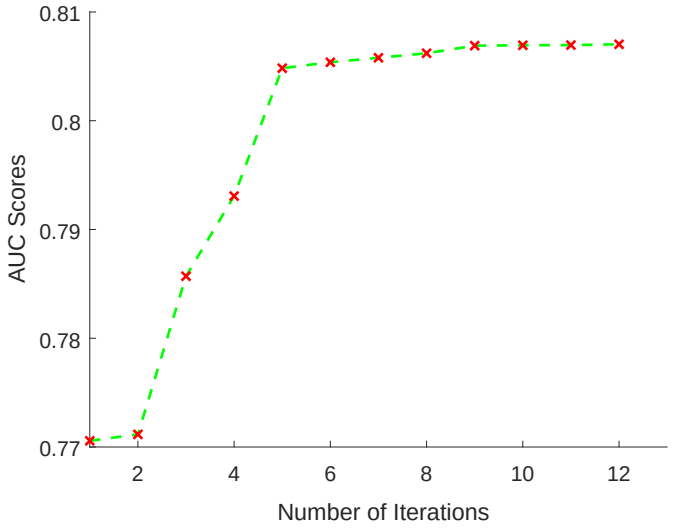


Figure 4.6.: AUC Scores: Running the Rescoring Model in Iterations

Furthermore, Figure 4.7 shows the results of running the model in 12^{th} iterations, where the detector output is also shown. As presented, the scores are changed in the 1^{st} run, but even improved considerably in the 12^{th} iteration run. `Zebra` objects scores have increased, where one instance reaches 1, which is likely to occur due to the context presented and the other detected objects presented. However, `elephant` class score is dropped considerably as in the 1^{st} run, and even more in the 12^{th} iteration, where this object is detected incorrectly.

Due to the success obtained during running the model in iterations, examining whether the model can be an effective tool in re-rating out-of-context objects. The answer, as shown Figure 4.8, is again a **Yes**. In this figure, only the outputs from Faster RCNN and the out-of-context model are shown, as the aim is to compare such outputs. Starting from the first row, we can see how the model decreases the confidences of `cat` due to its unlike location and scale. However, it can be seen how `cars` and `persons` confidences increase apart from `cars` that overlap with the `cat` (*e.g.*, the car next to the cat head), which are lower in scores compared to other cars. I assume this is because the detector is 2D based and having a car in such location compared with the cat is unlikely.[1] This is a great result that the model shows, which can be said that this model is a good tool for out-of-context objects.

Results in the second row also show how the model is reducing the confidence of the `toothbrush` as it is unlikely to be presented in such location and among such objects, co-occurrence among the detected objects (*i.e.*, `toothbrush` and `horses`) is very low as shown in the matrix in Appendix A. In the third row, two `sheep` are detected with confidences higher than 0.9, but when the model is applied, confidences are significantly reduced, as such objects are unlikely to be seen in location (*i.e.*, living room).[2]

Finally, for the last row, the model reduces the confidence of the `elephant`, which I suggest is due to the presence of the `couch`. This may raise a question, which is that *why only the*

[1] Image is taken from Instagram with permission from account @fransditaa.
[2] Images are made up using Google Images.

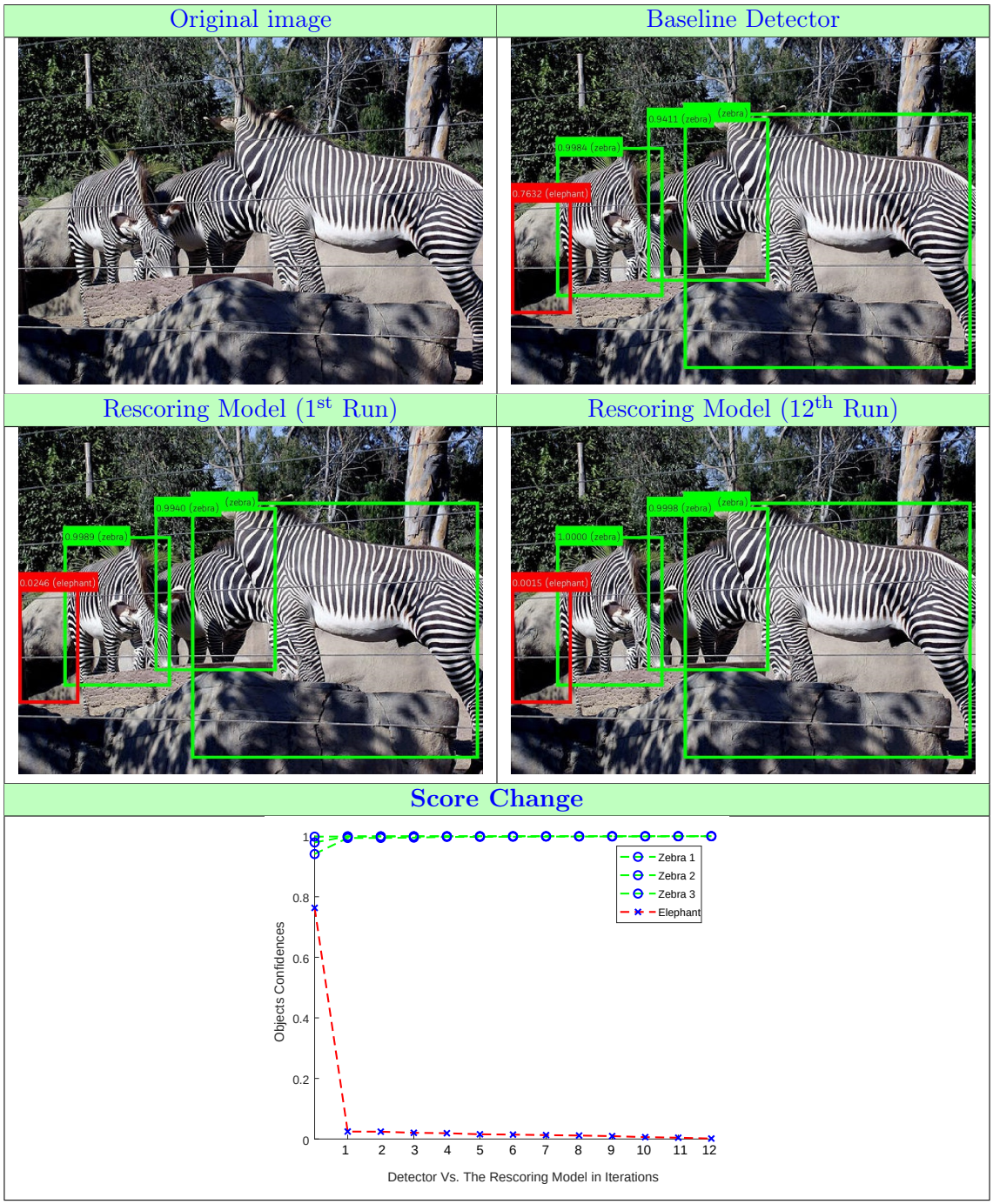


Figure 4.7.: Results: Running the Rescoring model in Iterations

`elephant` score is reduced, and not the `couch`, as `person` usually appears with both objects. I believe that the answer is that due to the location and the size of the `person` compared with the `couch`, the `elephant` is seen out-of-context in this scene, and this could be why the model reduces its confidence.[3]

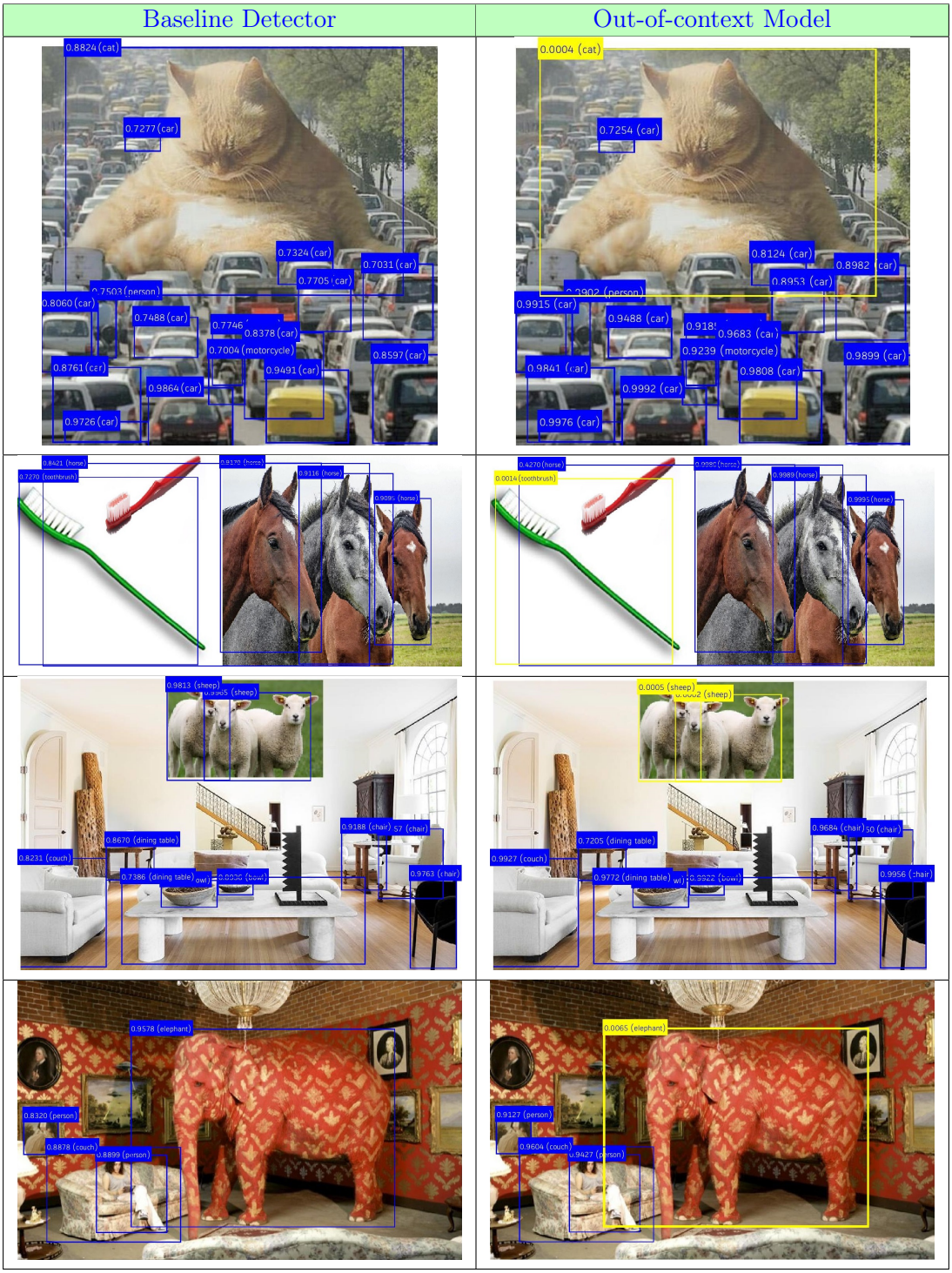


Figure 4.8.: Results: Out-of-Context, Yellow boxes represent out-of-context objects, blue boxes represent other objects regardless whether they are correctly detected or not.

[3]Image is taken from SUN dataset: Out-of-Context Images (Xiao et al., 2010).

Upon this success, I believe that the *Rescoring Model* (including the iterated model) is an effective tool to be used for re-rating objects scores upon the contextual knowledge observed from objects surrounding environments within the digital scenes. As shown (visually and statistically), the model outperforms the examined baseline detectors on various datasets, drawing the conclusion that such a model is capable of improving detectors outputs and be used as post-processing for most CNN-based detectors. This model is also improved and updated, to include the option for relabelling objects (not only re-rate scores), as presented in Section 4.3, below.

4.3. Relabelling

4.3.1. Experiment One: Relabelling Model

In this experiment, I researched further on how to improve the performance, and upon the success of the proposed contextual *Rescoring Model*, presented in Experiment Three 4.2.4, I then decided to move one step further to not only rescore object confidences but to also relabel them upon their contextual vicinity.

This is implemented as follows. *First*, I set a minimum threshold value for the contextual *Rescoring Model* as *0.4*. *Second*, any detected objects rescored by the *Rescoring Model* with less than the threshold are passed into the proposed pipeline, named the "***Relabelling Model***. *Third*, the top five possibilities obtained from the detector (only Faster RCNN is examined in this experiment) including the reference objects are passed into the *Rescoring Model*. If any of the possibilities are rescored with a higher value than the threshold, then the object(s) with the maximum value is considered as the new labelled object, if none is higher, then the reference object will be removed and considered as background. *Fourth*, after new labels are determined, all objects including the new labels, are passed again into the *Rescoring Model* again, to obtain the new confidences.

The proposed *Relabelling Model* is illustrated in Figure 4.9, where the process from inputting the images until outputted are shown. Note that all steps in the red square are the core processes involved in this approach.

Furthermore, *Relabelling Model*, as presented Table 4.18, obtain higher AUC scores than Faster RCNN and the *Rescoring Model*, in all threshold value cases, as three values of the threshold for Faster RCNN are used (*i.e.*, 0.7, 0.6 and 0.5). This is because the proposed *Relabelling Model* is not only re-rating objects confidences, but also suggesting new objects labels and removing objects with lower confidences than the set threshold value, based on the contextual information encoded from the digital scenes.

In addition, I use mean Average Precision (mAP), where IoU threshold is 0.5, and F1 score are used as additional evaluation metrics to show the effectiveness of the *Relabelling Model* (See Section 2.4, for more information about evaluation approaches). Results of using such evaluation metrics are presented in Table 4.19. The *Relabelling Model* clearly achieves a

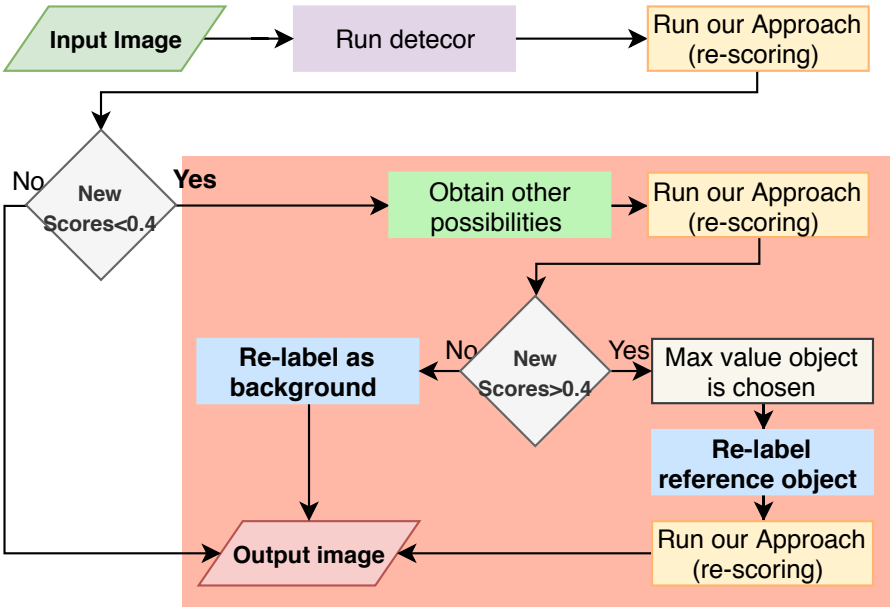


Figure 4.9.: Relabelling Approach

Table 4.18.: AUC Scores: Faster RCNN vs. Rescoring model and Relabelling Model.

Threshold Value	Faster RCNN	Rescoring Model	Relabelling model
0.7	0.76472	0.77057	**0.78278**
0.6	0.77911	0.78562	**0.79446**
0.5	0.79303	0.80423	**0.81084**

better performance than the baseline detector (*i.e.*, Faster RCNN) in terms of improving both mAP and Fl scores.

Table 4.20 shows some results of the *Relabelling Model* outputs. As three different threshold values (*i.e.*, 0.5, 0.6 and 0.7) are used, where one/or more results are shown per threshold value. Starting from the first row, where value of the threshold used for Faster RCNN is 0.5, the input image, detector, *Rescoring* and *Relabelling Models* outputs are presented. Three objects are detected, which are two `persons` and a `snowboard`, scored by the detector as *0.9992*, *0.8366*, and *0.5717*, where the second `person` and `snowboard` are incorrectly detected. The *Rescoring Model* is then applied, which changes the objects confidences as to *0.9406*, *0.3737* and *0.2500*, respectively. It can be noticed that all scores are reduced, particularly the scores of the incorrectly detected objects, which are considerably dropped. Since scores of such objects are rated with less than 0.4 (the threshold value set for the *Relabelling Model*), the *Relabelling Model* is run, which labels the second `person` as background and changes the `snowboard` to a `skis` making them as correct detection, where

Table 4.19.: mAP and F1 scores in percentages [%]: Faster RCNN vs. Relabelling Model.

Threshold Value	Faster RCNN		Relabelling Model	
	$mAP_{0.5}$	F1	$mAP_{0.5}$	F1
0.7	62.82	57.34	**65.50**	**58.95**
0.6	57.55	52.77	**64.14**	**56.35**
0.5	51.38	48.68	**63.14**	**55.02**

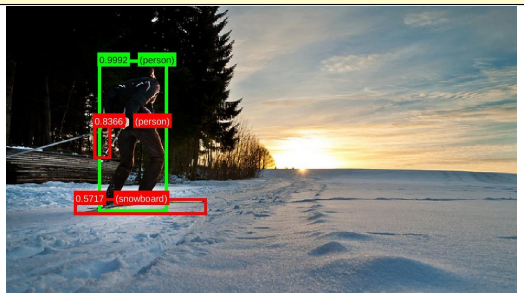
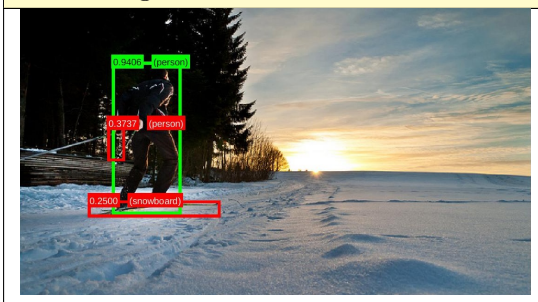
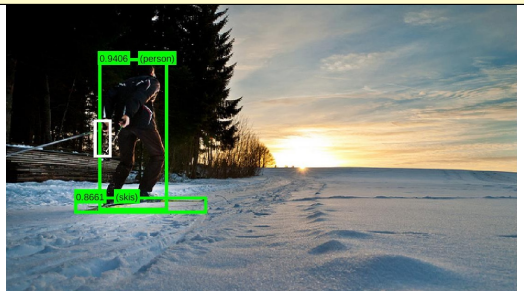
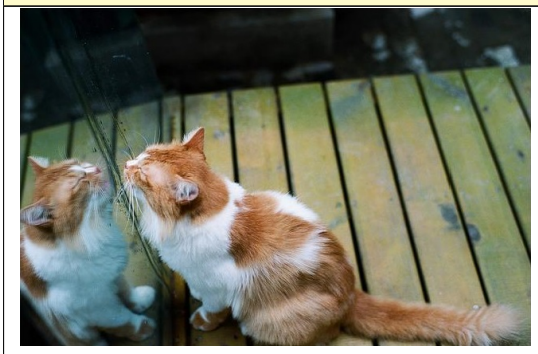
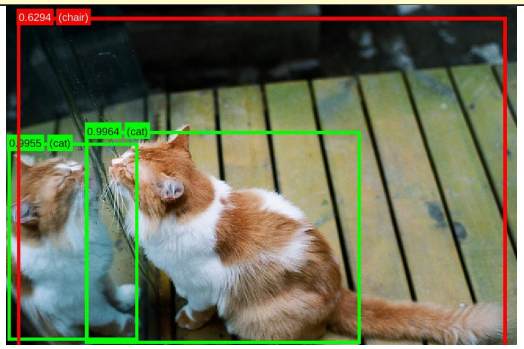
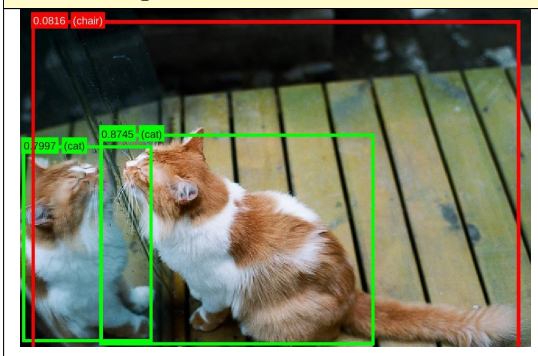
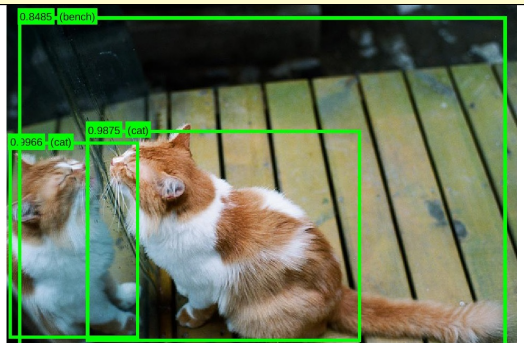
scores also changed, as for `person` to be *0.9406*, and `skis` as *0.8661*.

Threshold value for the detector is changed to 0.6, as presented in the second tested images. The *Relabelling Model* corrects the label for one of the detected object, which is changed from a `chair` to a `bench`. In detail, the detector is first run on an image detecting two `cats` and a `bench`. The detector detects the two `cats` correctly, with confidences of *0.9955* and *0.9964*, but incorrectly detects the `bench` as a `chair` with a confidence of *0.6294*. The *Rescoring Model* is applied, which decreases the `chair` scores to *0.0816*, making it ready for the *Relabelling Model* to figure out if there are another object that is more likely to fit. The *Relabelling Model* improves the detection, which suggests the replacement of a `bench` instead of the `chair` with confidence of *0.8485*, which therefore, leads to an increase in other objects scores. Another result obtained from running the detector with the same threshold value, where two `cakes`, a `person` and a `bowl` are detected. The detector detects the `cakes` as `sandwiches`, but when the *Relabelling Model* is applied, it corrects them to `cakes`. However, the *Relabelling Model* still fails to remove the `person`, which is incorrectly detected, but still it reduces its scores from *0.7856* to *0.4078*. This is because it is rated by the *Rescoring Model* with higher than 0.4, and thus looking at other possibilities is not applicable.

Detector with a threshold of 0.7, as the main baseline detector threshold of all experiments in this work, is run as well. As seen in the first row of images in the threshold 0.7 section that the *Relabelling Model* processes the inputted image very well, where it correctly the relabels of three incorrectly detected objects, which are two `carrots` and a `book`. A `knife` is first incorrectly detected as a `book` by the detector, but when the *Rescoring Model* rates as *0.2149*, the *Relabelling Model* relabels as a `knife` with confidence of *0.7895*, which is more likely in such a context. Similarly, in the following results, we can see how to the *Relabelling Model* removes the incorrectly detected objects (*i.e.*, `traffic light`), and also increases the confidences of the correct detected objects. I believe this great improvement is achieved by the model due to the spatial and scale relationships among the detected objects. In other words, `traffic lights` are unlikely to be smaller than a `person` and overlapped in such a way, and because the detector did not suggest a better fitting for such objects in such location, the *Relabelling Model* suggests relabelling as background, which results to correct detection.

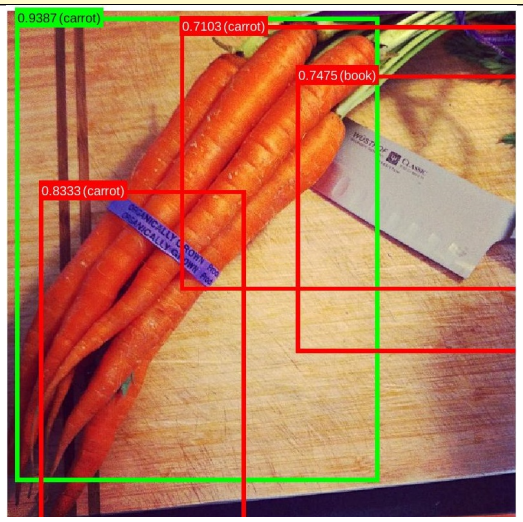
However, the *Relabelling Model* can also lead to negative relabelling. As illustrated in the last result, we can see that the *Relabelling Model* suggests a `kite` instead of a `surfboard`, making this incorrect detection, where it also reduces all objects scores including the correctly detected objects.

Table 4.20.: Relabelling and Rescoring Models outputs: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively

Threshold Value is 0.5	
Original image	Baseline Detector
	
Rescoring Model	Relabelling Model
	
Threshold Value is 0.6	
Original image	Baseline Detector
	
Rescoring Model	Relabelling Model
	

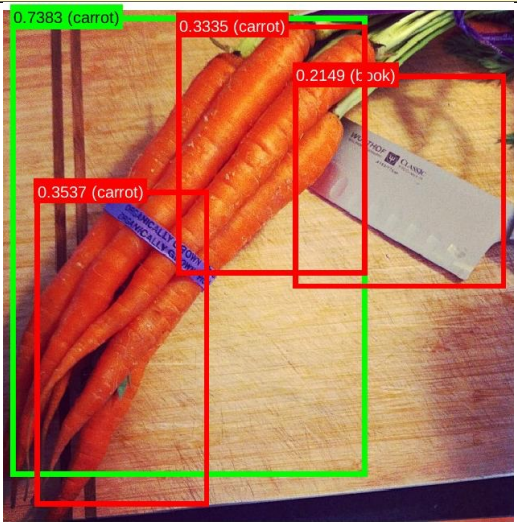
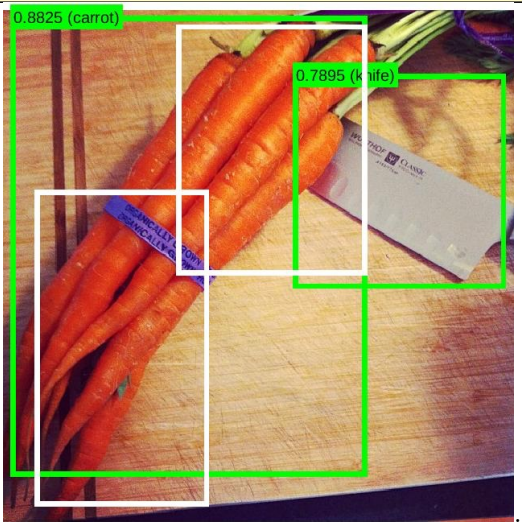
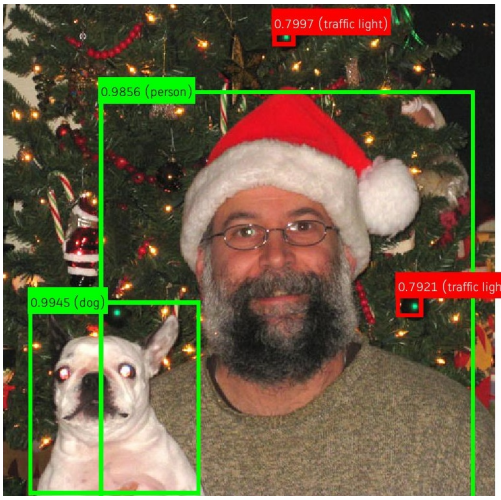
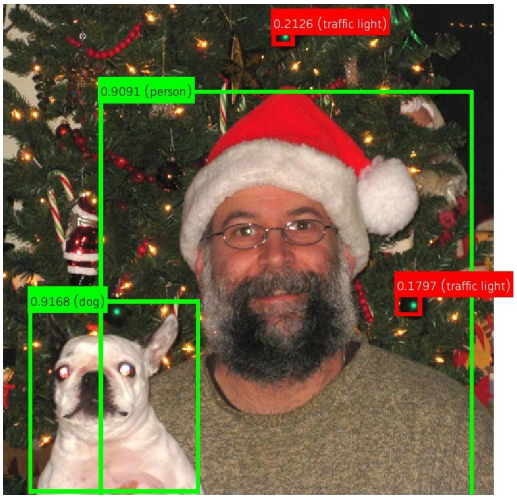
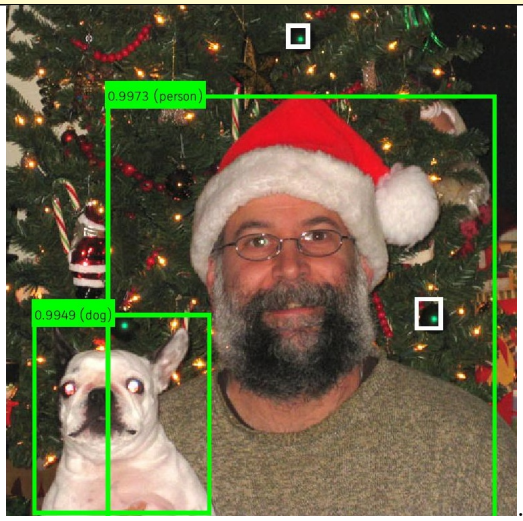
Continued on next page

Table 4.20 – continued from previous page

Threshold Value is 0.6	
Original image	Baseline Detector
	
Rescoring Model	Relabelling Model
	
Threshold Value is 0.7	
Original image	Baseline Detector
	

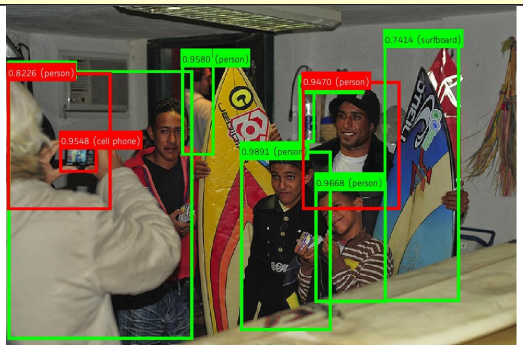
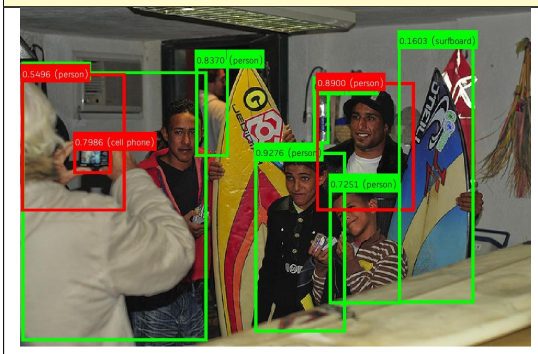
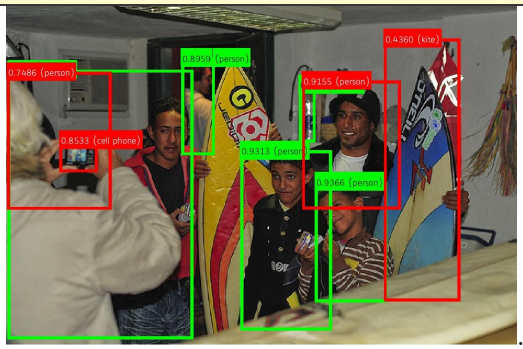
Continued on next page

Table 4.20 – continued from previous page

Rescoring Model	Relabelling Model
	
Threshold Value is 0.7	
Original image	Baseline Detector
	
Rescoring Model	Relabelling Model
	

Continued on next page

Table 4.20 – continued from previous page

Threshold Value is 0.7	
Original image	Baseline Detector
	
Rescoring Model	Relabelling Model
	

Furthermore, due to the high number of object classes in MSCOCO, only nine randomly chosen objects AUC scores are presented in Table 4.21, to illustrate the comparison between Faster RCNN, the *Rescoring Model* and the *Relabelling model*. Objects names, objects IDs as annotated in MSCOCO 2017, the AUC scores per class are shown in terms of all three models are presented. The *Relabelling model* has the majority of objects with the highest scores, resulting in the best performance as the average of AUC scores with *0.7827*. The reader is referred to Appendix D.1, for all objects AUC scores. As previously done, the *Rescoring Model* is iterated leading to better results, upon this iteration, I apply the *Relabelling Model* aiming to obtain better performance, as illustrated in Section 4.3.2.

The *Relabelling Model* is also examined in comparison to one of the state-of-the-art contextual models, named Object Relation Module (ORM) proposed by Hu et al. (2018), which was discussed in Section 3.4. Both the *Relabelling Model* and ORM benefit from the use of pre-defined relationships, which are explicitly encoded from the scene. However, *Relabelling Model* uses sixteen relationships, where ORM uses only four, which include only the spatial and scale contexts. *Relabelling Model*, therefore, has the semantic context included as well. ORM is a fully end-to-end object detector, whereas *Relabelling Model* is used as a post-processing step that can be applied upon the use of both one-stage and two-stage detectors. Both models, as presented in Table 4.22, are examined on MSCOCO 2017 dataset (only on images with more than one object detected), where mAP and F1 scores for both models, and some features are reported.

Table 4.21.: AUC scores for some MSCOCO object classes: Faster RCNN, Rescoring and Relabelling Models.

Class ID	Class Label	Faster RCNN	The Rescoring	The Relabelling
1	Person	0.84345	**0.84646**	0.84568
2	Bicycle	0.76073	0.76984	**0.78006**
10	Traffic light	0.78979	0.78690	**0.79083**
18	Horse	**0.92223**	0.89649	0.88535
:	:	:	:	:
34	Kite	0.73866	0.76852	**0.78292**
:	:	:	:	:
43	Fork	0.66847	0.79619	**0.81545**
44	Knife	**0.69595**	0.59494	0.56944
:	:	:	:	:
72	Sink	0.76167	0.78766	**0.79756**
78	Teddy bear	**0.80049**	0.78015	0.77897
Mean	-	0.76472	0.77057	**0.78278**

Table 4.22.: A comparison between the Relabelling Model and Object Relation Module. mAP and F1 scores in percentages [%] are reported as evaluated on MSCOCO 2017.

Features	Relabelling Model	Object Relation Module
End-to-End Model		✓
Post-Processing	✓	
Applied to one-stage detectors	✓	
Applied to two-stage detectors	✓	✓
Semantic context	Explicit	Implicit
Spatial context	Explicit	Explicit
Scale context	Explicit	Explicit
No. Relationships	16	4
mAP	34.07	33.81
F1	58.95	58.16

4.3.2. Experiment Two: Iterated Relabelling Model

Upon the success that the *Relabelling Model* shows (Section 4.3.1), it is expected to obtain a much higher performance when applying the *Relabelling Model* on the top of the *Iterated Rescoring Model* due to the good performance it shows (Section 4.2.4), compared to the *Rescoring Model* with no iteration (Section 4.2.4). As presented in Table 4.17, the *Rescoring Model* performance increases when iterated, until the 12^{th} iteration, due to this great performance, applying the *Relabelling Model* on the 12^{th} *Iterated Rescoring Model*, named *Iterated Relabelling model (IRM)* is expected to perform greater than when applying on the plain *Rescoring Model*.

For the statistical analysis, Table 4.23 shows the average AUC scores comparing the performance of the baseline detector (Faster RCNN) with the *Iterated Relabelling Model*. It can be seen that the *IRM* shows a very great increased performance, which is expected, as it corrects labels and changes object confidences. Many of the objects AUC scores reaches one, meaning that it is almost sure about the object being correct. The reader is referred to Appendix D.2 for all objects AUC scores.

Table 4.23.: A comparison between ten MSCOCO object classes AUC scores for Faster RCNN (threshold is 0.7) and the Iterated Relabelling Model.

Class ID	Class Label	Faster RCNN	IRM
1	Person	0.84345	**0.99955**
2	Bicycle	0.76073	**0.99170**
:	:	:	:
5	Airplane	0.88300	**1**
21	Elephant	0.74058	**0.96282**
25	Backpack	0.61494	**1**
27	Handbag	0.67471	**1**
:	:	:	:
37	Skateboard	0.84024	**0.97560**
44	Knife	0.69595	**0.97878**
45	Spoon	**0.65490**	0.57254
:	:	:	:
67	Keyboard	0.75925	**0.96692**
Mean	-	0.76472	**0.95314**

Table 4.24.: AP and F1 scores in percentages [%] for Faster RCNN and the Relabelling model.

Model	$mAP_{0.5}$	mAP	F1
Faster RCNN	62.82	33.48	57.34
Relabelling Model	65.50	34.07	58.95
Iterated Relabelling Model	**70.10**	**40.43**	**64.84**

In addition, not only the AUC scores used to evaluate this model, but, also F1, $mAP_{0.5}$ and mAP (IoU=[0.5:0.05:0.95]) scores are applied, as illustrated in Table 4.24. This Table shows the F1, and mAP scores in percentage for Faster RCNN, the *Relabelling Model* and the *Iterated Relabelling Model*, where the threshold used for the detector is 0.7. As noticed, *IRM* obtains the highest scores with an improvement of 8% is achieved compared to the detector performance.

Results obtained from the *IRM* is visualised as in Figure 4.10. As the comparison in this figure is intended only to compare the outputs of the detector (*i.e.*, Faster RCNN) and *IRM*, only their outputs are shown. The results obtained from Faster RCNN and *IRM* are shown in the left and right columns, respectively. In the first row, three objects are detected, which are a `person`, a `sports ball`, and a `tennis racket`. All objects are correctly detected with confidences higher than 0.94 in all cases. Once inputted into the *IRM*, the model even increases the performance of each object with a minimum of 0.98. This can be due to the high semantic, spatial and scale relationships. They are likely to appear in real-world in such position and scale.

In the second row, five objects are detected: three are correctly detected, which are a `person` and two `elephants` and the other two are incorrect, which are a `person` and a `cow`. Even-though, Faster RCNN detects all objects with high confidence including the incorrect objects (*i.e.*, minimum confidence is 0.74 for the `cow`), the *IRM* discards all the incorrect objects and label them as background, whereas, increases the correctly detected objects confidence. I believe this is due to the spatial and scale relationships between `person`

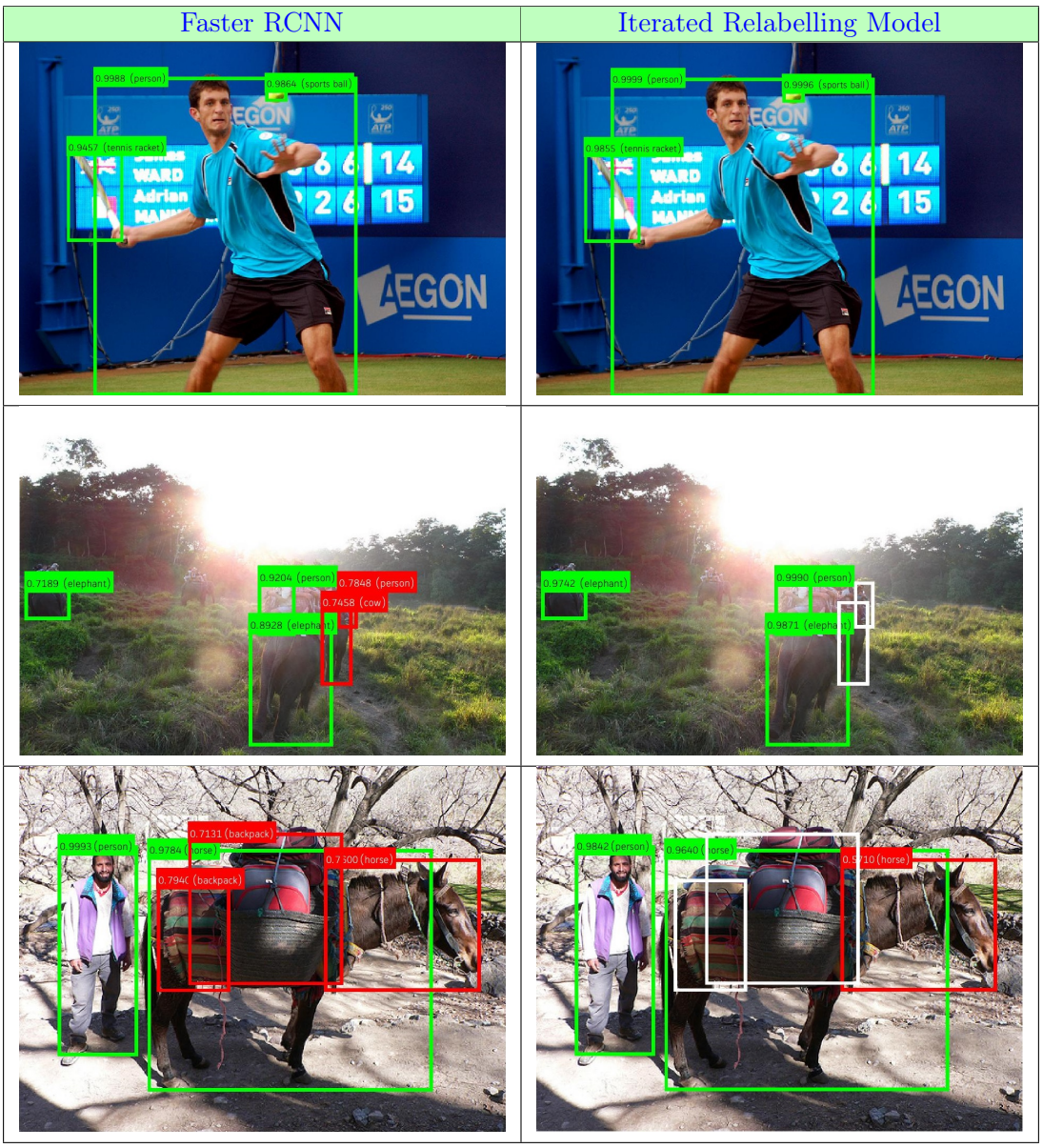


Figure 4.10.: Results: Faster RCNN vs IRM outputs: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively

and `elephant`. This can also be due to the presence of two `elephants`, which, therefore, reduces the likelihood of a `cow`, particularly because it overlaps with one of the `elephants`, in an unlikely scale (*i.e.*, small boundary boxes in respect of other objects).

Finally, in the third row, five objects again are detected, but 3 are incorrect. Only two of the three incorrect objects are labelled as background by the *IRM*, where the third object's confidence (*i.e.*, incorrect `horse`) were dropped from *0.7600* to *0.5710*. Due to the complexity of this scene and the high number of incorrect detected objects, the *IRM* also decreases the confidences of the correct objects (*i.e.*, `person` and `horse`) from *0.9993*, *0.9784* to *0.9842*, *0.9640*, respectively, but as said, it is always hard to find the perfect model that can solve all issues.

4.4. Summary

This chapter presented how contextual information is extracted from the baseline detectors and passed into the contextual models. Two proposed contextual models named the *Rescoring Model* and the *Relabelling Model*, were applied as post-processing to most of the detectors aiming to improve the confidence scores of the detected objects upon the objects contextual knowledge with others rather than their appearance features. The *Relabelling Model* also steps further to not only re-rate objects scores, but also relabel objects (*e.g.*, change incorrect detection to correct, to some extent) benefiting from observing the contextual information in the processed digital images.

The proposed models have higher average AUC scores compared to the state-of-the-art baseline detectors (Faster RCNN and YOLOv1). Moreover, they are run in iteration, which leads to an improvement also in terms of mAP. Some of the advantages these models have are as follows:

1. They improve the detection performance of the state-of-the-art object detectors due to the inclusion of semantic, spatial and scale contexts, which are observed to be effective in this regards.

2. They can be integrated as post-processing to most CNNs-based object recognition frameworks, whether detectors are one-stage or two-stages. This is in contrast to Redmon et al. (2015), which is specific to their end-to-end pipeline, and Hu et al. (2018), which can only be applied on two-stage detectors. See Chapter 2 for more details about CNN-based detectors.

3. Rather than only evaluating whether the detected regions are correctly detected as in Choi et al. (2012b) and Cao et al. (2015), the proposed models step further to rescore and relabel detections.

4. They are built upon the use of sixteen contextual relationships, as presented in Chapter 3, unlike other models that use a smaller number of relationships such as Choi et al. (2012b); Chen et al. (2013); Barnea and Ben-Shahar (2018); Bozcan and Kalkan (2019).

5. Compared to Inoue et al. (2017); Mottaghi et al. (2014); Cinbis and Sclaroff (2012), which mainly rescore objects probabilities, the proposed relabelling model steps ahead to relabel objects and rescore objects based on their relationships with other detected objects.

In short, this section showed that semantic, spatial and scale relationships enhance the detection performance, where correcting and relabelling false detection can also be attempted. Rather than only examining the effectiveness of the contextual knowledge among objects on real-world images, perturbed images are used. Using such type of images can clearly examine the impact of the proposed models, as the majority of detectors fail to have good performance when such images are inputted. This issue is discussed in detail, below, in Chapter 5.

Chapter 5

IMPACT OF CONTEXT ON PERTURBATIONS

5.1. Overview

CNN-based detectors have shown great performance in classifying, detecting, localising, and recognising objects within digital images. However, it was found that adding certain perceptible perturbations, which can be quasi-imperceptible to humans, can misclassify image labels and increase the prediction error rate, leading to drop detectors performance significantly (Szegedy et al., 2014b). In this chapter, the application of applying perturbation, named "*adversarial perturbations*", on images is discussed due to the importance of this issue including the risks and the impacts that they cause in reducing detectors performance. Reducing the impact of this application is still ongoing research, several studies attempt to address this issue, but none are seen optimal yet (Goodfellow et al., 2014; Papernot et al., 2015, 2016).

Therefore, due to the success that the contextual *Relabelling Model* (presented in Chapter 4) shows in improving detection performance. It is proposed, here, to reduce the impact of the attack, and improves detector competence after being attacked (*i.e.*, perturbations are added to inputted images). The baseline detector used in this chapter is Faster RCNN, trained and evaluated on MSCOCO 2017 dataset.

This chapter presents the definition of adversarial perturbations, and how several studies have attempted to generate them. Experiments are presented illustrating the efficiency of using the *Relabelling Model* in improving the performance of the detector when attacked, where two types of attacks are applied, which are Universal Adversarial Perturbation (UAP) (Moosavi-Dezfooli et al., 2016) and Fast Feature Fool (FFF) (Mopuri et al., 2017).

Figure 5.1.: Impact of UAP Adversarial Perturbations On Inception Classifier

5.2. Background

In this section, the definition of adversarial perturbations and how they can be generated are presented. Later, some studies proposing some solutions to address this issue are discussed and explored.

5.2.1. What is Meant by Adversarial Perturbations?

Adversarial Perturbations are meant to be some noise (not random noise) but rather carefully computed function added to images and be expecting to fool the Deep Neural Networks (DNNs) to misclassify image labels. DNNs are expected to be inputted with an image and predict its contents. However, when perturbations are added to inputted images, DNN models predictions are seen to fail and predict incorrect labels with higher confidences.

As shown in Figure 5.1, two images are presented, which are the classifier output before and after perturbations added, as on the left and right sides respectively.[1] Yes, it can be said that both images may seem the same from a human vision system perception, whereas, machine sees them different. As illustrated, the classifier classifies the object as `trolleybus` (correct prediction) before perturbations added (left image), but misclassifies the object (*i.e.*, predicted as `thimble`; incorrect detection), after perturbations added.

In short, this mere addition of perturbations fools the network and drops its performance resulting to an increase in the fooling rate. This is in line with Szegedy et al. (2014b), who claim that the adversarial examples are so close to the original examples, where the differences maybe indistinguishable to the human eye. I hypothesise that including objects contextual information can help to improve the detectors performance, thus, this chapter

[1] The results shown are obtained using Moosavi-Dezfooli et al. (2016) Python Code, which is available at https://github.com/LTS4/universal, where the image tested is taken from MSCOCO 2017 validation dataset. The classifier used is inception5h, trained on Imagenet

is to test and explore this hypothesis. This apparent weakness is in contrast to how robust human perception of objects is to a variety of noise and perturbations. One plausible reason for this difference in robustness between human and artificial perception is context, thus contextual model is suggested.

5.2.2. How Are Adversarial Perturbations Generated?

Adversarial Perturbations are defined, above, as some mere additions to the images, but finding a solution to minimise its effects requires understanding how they are generated. Szegedy et al. (2014b) state that adversarial perturbations are some small changes in the images in respect of how DNNs see images compared to the original images. They induce classifiers to change their predictions. Generating adversarial examples is considered as an optimisation issue, hence, optimisation algorithms are used to develop such examples. Szegedy et al. (2014b) generate adversarial perturbations using the box-constrained limited-memory approximations of Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimisation algorithm. However, such an approach is seen to consume high computational cost, thus Goodfellow et al. (2014) propose another method called Fast Gradient Sign (FGS). FGS method is considered to develop perturbations using the sign of the gradient of loss in respect to the input, where the gradient is computed using back-propagation. In their paper, it is reported that the proposed method impacts the classifiers and change their predictions to misclassify objects. Even though, both L-BFGS and FGS use gradient information, FGS is seen much faster because the gradient is used only once as an explicit direction for a line search. However, L-BFGS performs several gradient computations.

Furthermore, Moosavi-Dezfooli et al. (2016) propose a single perturbation image, called *Universal Adversarial Perturbation (UAP)*. UAP is reported to fool several networks, which are trained on the same data, with high probability. As claimed, UAP is "*quasi-imperceptible universal perturbation vectors that lead to misclassify natural images with high probability*" (Moosavi-Dezfooli et al., 2016). However, a drawback of this method is that it requires access to the training data that the network is initially trained on. Mopuri et al. (2017), therefore, claim that due to this drawback, they propose another approach that is seen to address this shortcoming. The proposed method, as stated, is a novel data independent method, which compute universal adversarial perturbations, named *Fast Feature Fool (FFF)*. FFF does not require any prior knowledge to the data used of the applied model structure. As claimed, "*a single perturbation that can fool a target CNN on most of the images without any knowledge of the target data*" including *"type of data distribution (e.g., faces, objects, scenes, etc.), number of classes, sample images"* is not needed (Mopuri et al., 2017).

In this work, both UAP and FFF are adapted and used for examining Faster RCNN performance and the proposed *Relabelling Model*.

5.2.3. Are Adversarial Perturbations Transferable?

One of the most important question that this chapter answers is are *Are Adversarial Perturbations Transferable?*. The answer can be found in this section.

As Carrara et al. (2019) state that adversarial examples can exhibit cross-model and cross-training, where the same adversarial perturbations can also impact different models with different architectures. In line with this, Szegedy et al. (2014b) state that the same adversarial example is often misclassified by a variety of classifiers with different architectures or trained on different subsets of the training data. Adversarial examples can be also transferred to real world. This is claimed by Kurakin et al. (2016) after they added perturbations to images, they printed them, then recapture them using a phone camera. Recaptured images are then inputted into a classifier. It is found that the classier misclassifies objects presented in the recaptured images.

In addition, Moosavi-Dezfooli et al. (2016) test their adversarial perturbation approach (UAP) on various classifiers (e.g. CaffeNet (Jia et al., 2014), VGG-F (Chatfield et al., 2014), VGG-16 (Simonyan and Zisserman, 2015), VGG-19 (Szegedy et al., 2014a)), and claim that the addition of the perturbations drops the tested classifiers' performance by almost 90% as tested on ImageNet validation dataset. Therefore, this leads to the result that single universal perturbation vectors can cause most of the natural images to be misclassified. They also add UAP to some images taken by mobile phone and find that they are misclassified, showing that UAP is "*powerful image-agnostic perturbation" [that can fool] "any image"* even with high classification confidence (Moosavi-Dezfooli et al., 2016). Therefore, it can be concluded that using UAP perturbations computed on the X network architecture is likely to misclassify images testing with Y network architecture, as they show how perturbations computed (for example) on VGG-F and tested on another classifier (e.g. CaffeNet), but both models are conditional to be trained on the same dataset (ImageNet).

However, Mopuri et al. (2017) claim that their proposed model FFF is transferable various different models (*i.e.*, they use the same models used in Moosavi-Dezfooli et al. (2016)). The main and important difference between UAP and FFF is that FFF is data-independent. In other words, it can be applied on any models even if the trained and tested models are trained on different datasets (*i.e.*, no prior knowledge about the target CNNs is needed). It is also reported that when FFF is trained on X dataset, and evaluated on Y dataset, it impacts the CNN models, leading to an increase in the fooling rate compared to UAP, which is data-dependent method.

Due to the impact that both methods (*i.e.*, UAP and FFF) cause to the CNN models, they are used in this chapter to examine the impact of the contextual *Relabelling Model*. UAP perturbation used in experiments is trained on MSCOCO 2017, as it is data-dependent. However, FFF is used as proposed (*i.e.*, already trained on ImageNet), but will be tested on MSCOCO 2017.

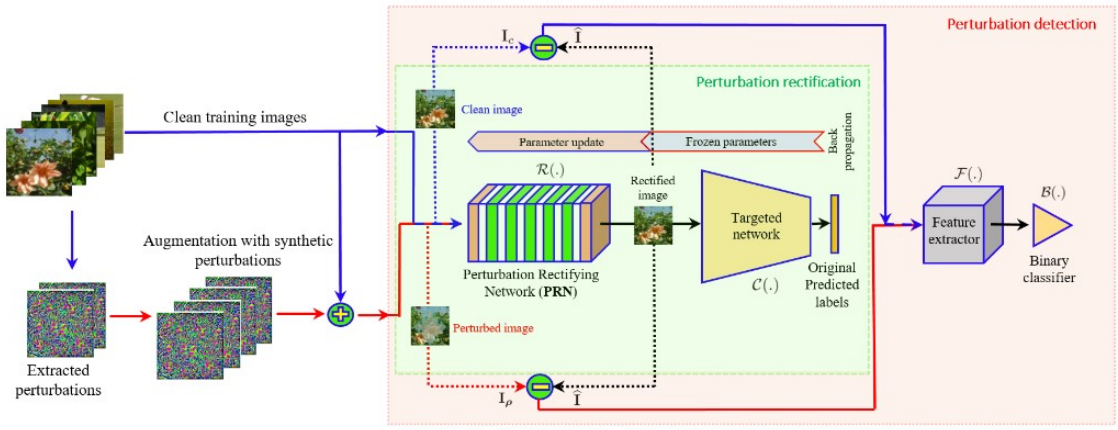


Figure 5.2.: PRN processes, adapted from Akhtar et al. (2018)

5.2.4. Adversarial Perturbations Defences

As already stated, adversarial attacks pose a serious security threat to DNNs and particularly real-world applications such as autonomous cars, biometric identification, and surveillance. Thus, mitigating their impact can be seen as a must. This section discusses some methods defending models against adversarial attacks. Such methods can be grouped into two categories, which are i) to pre-process images through various image processing algorithms before passed into the DNNs, and ii) to train DNN models through adversarial training to enhance robustness (Chaubey et al., 2020).

Akhtar et al. (2018) propose a framework to defend DNN networks against perturbations. The proposed approach consists of a network, named Perturbation Rectifying Network (PRN), used as pre-input layers to the models. This means that the targeted models do not require any modification. The PRN pre-processes the input image before passing it to the target model. It is trained on real and synthetic image-agnostic perturbations. In other words, an image is first passed to the PRN, if a perturbation is detected, the output of the PRN is used for label prediction instead of the actual image. PRN works to subtract the perturbation from the perturbed image, to restore original image, by the use of binary classifier called (by the authors) as the Detector Network. Furthermore, Figure 5.2 presents the whole process applied in this model showing the all stages applied in the PRN and the detector network. This model was evaluated in comparison with some classifiers (*i.e.*, CaffeNet (Jia et al., 2014) and VGG-F (Chatfield et al., 2014) resulting to up to 97.5% success rate.

In terms of the other category of defence, which is via using adversarial training, Shafahi et al. (2018) propose two methods of universal adversarial training. The first is to model the training process as a two-player min-max game where the minimisation is over the target model parameters, and the maximisation is over the universal adversarial perturbation. This method is seen as time-consuming (*i.e.*, 2× the cost of natural training). However, the second method, which reduces the training time by half as compared to the previous one, improves the defence efficiency by providing a 'low-cost' algorithm with a slight decrease in robustness. Both methods were evaluated on ImageNet, leading to an improvement in

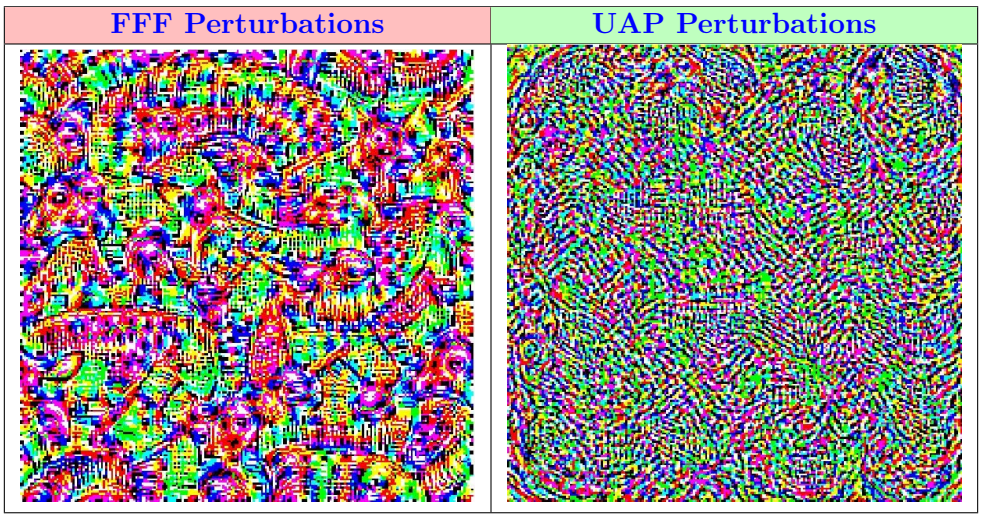


Figure 5.3.: FFF vs. UAP Perturbations added to images, size=[224×224].

the performance, mitigating the impact of adversarial perturbations.

While adversarial training shows an improvement in the performance, and has been investigated in various studies (*e.g.*, Mummadi et al. (2019); Ren et al. (2020)), it still leaves models sensitive to other adversarial perturbations that were not included in the training phase, plus it can be seen as time-consuming. Therefore, leveraging contextual information presenting on images (natural and perturbed) can be yet a solution, as it can be applied to any DNN models with no modifications needed, as examined in Section 5.3.

5.3. Experiments

In this section, I present in depth some experiments I attempt to examine the impact of adversarial examples on Faster RCNN detector using MSCOCO 2017 dataset, and how the contextual model can help to improve the performance. Two types of adversarial examples are used, which are UAP, and FFF. Figure 5.3 shows the perturbations sized 224×224, which are added on images. The left image shows the FFF perturbations, whereas on the right UAP perturbations are presented. It can be seen how both look different to each other, even though they both are trained on the same model (*i.e.*, VGG-16), but on different training dataset, which are ImageNet for FFF, and MSCOCO 2017 for UAP. Perturbations, moreover, can be added on the entire image, and on regions, both options are examined as illustrated in experiments below.

5.3.1. Experiment One: Perturbation On The Entire Image

In this experiment, the impact of adversarial perturbations on the entire image is illustrated in terms of both Faster RCNN detector and the *Relabelling Model*. As shown in Figure 5.4, I am following the steps below:

1. Input an image from MSCOCO 2017.

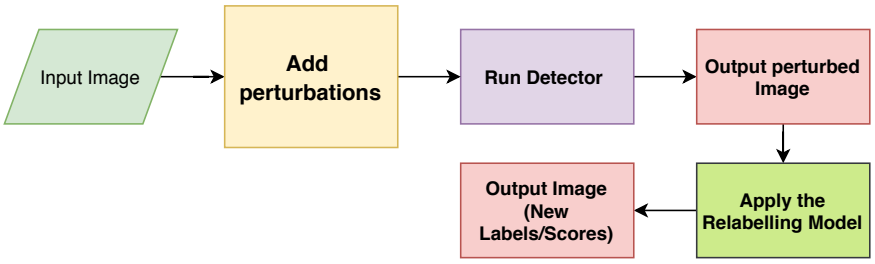


Figure 5.4.: Apply perturbations on the entire Image

2. Add perturbations to the inputted image.
3. Input the perturbed image into Faster RCNN.
4. Run Faster RCNN.
5. Obtain detector outputs (*i.e.*, bounding boxes, labels and confidence).
6. Apply the contextual *Relabelling Model* to obtain new scores/labels.

As stated, two types of adversarial perturbations are used in this experiment (*i.e.*, UAP and FFF). Both of which significantly drop the performance of several models, and thus they are tested in this section. The main difference between both models is that UAP is considered as data-dependent, and FFF is data-independent. Therefore, in this section UAP has been trained on MSCOCO 2017 using VGG16 (*i.e.*, as used in Faster RCNN), whereas FFF is used as it is proposed (trained on ImageNet) with VGG16 model. MSCOCO 2017 validation dataset is used (*i.e.*, only images with more than one object detected, as images naturally appear, are used).

Impact of Perturbations

The aim of this experiment is to examine to what extent the contextual *Relabelling Model* helps the chosen detector in improving its performance and reducing the error rate when adversarial perturbations are added to image. However, illustrating the performance of the detector when adversarial perturbations are added seems a need as the *Relabelling Model* works as post-processing step. This is because later in this work, only the outputs of the detector and the contextual model on perturbed images are shown, for a visualisation simplicity purposes.

In Figure 5.5, the inputted image is shown before and after attack, following by the detector and the *Relabelling Model* outputs in both cases. The type of attack used in this example is UAP. It can be seen that the image is inputted and passed into the detector and then the *Relabelling Model*. In comparison between the detector performance before and after the attack, it can observed that only two classes are detected after perturbation is applied out of five detected before. This can illustrate to what extent does this attack impacts the detector performance. The contextual *Relabelling Model* is applied, which increases object confidences, but makes no changes in labels as both detected objects are correctly

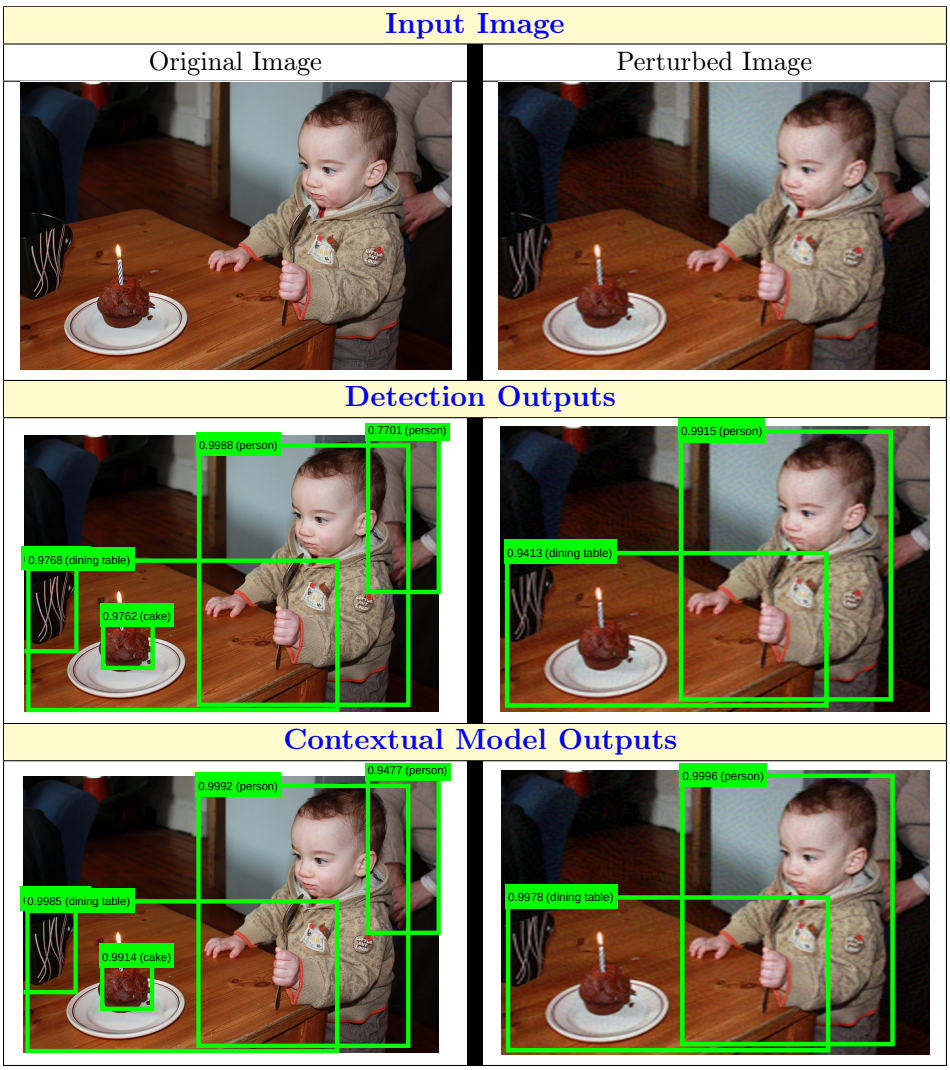


Figure 5.5.: Comparison between Detection and Relabelling Model outputs before and after perturbations applied

detected. Below, the statistical analysis and more visual results are shown, and discussed.

Table 5.1.: mAP and F1 scores in percentages [%] for the Faster RCNN and the Relabelling Model on perturbed images.

	FFF		UAP	
Models	$mAP_{0.5}$	F1 Score	$mAP_{0.5}$	F1 Score
Faster RCNN	45.34%	39.13%	46.40%	40.05%
Relabelling Model	**49.39%**	**41.91%**	**50.72%**	**43.12%**

Table 5.2.: A comparison between some MSCOCO object classes AUC scores between Faster RCNN and Relabelling Model on perturbed images.

	FFF		UAP	
Class Label	Faster RCNN	Relabelling Model	Faster RCNN	Relabelling Model
Person	0.62389	**0.69068**	0.62881	**0.69692**
Bicycle	0.44476	**0.51329**	0.49582	**0.60509**
Train	0.42961	**0.47233**	0.41062	**0.50656**
Boat	0.24063	**0.33439**	0.22854	**0.363888**
Fire Hydrant	**0.61949**	0.54481	**0.53459**	0.38207
Stop Sign	0.76242	**0.77329**	0.69155	**0.71428**
Cow	**0.30345**	0.29685	0.29946	**0.33333**
Bear	0.24074	**0.27778**	0.15	**0.31667**
Kite	0.15154	**0.27054**	0.14601	**0.20830**
Surfboard	0.39266	**0.50341**	0.44750	**0.52104**
Fork	0.39130	**0.52391**	0.28726	**0.41770**
Pizza	0.67393	**0.78464**	0.63185	**0.75875**
Plotted Plant	0.29679	**0.36531**	0.27156	**0.36419**
Toothbrush	**0.4**	0.3636	**0.2**	0.18181
Mean	0.34714	**0.42030**	0.36050	**0.43342**

Performance of Models On Perturbed Images

Perturbations are added to the entire image before inputted into Faster RCNN, it is observed that the performance of the detector drops by $\approx$ 17 points (*i.e.*, mAP for Faster RCNN is 62.82% before attack, and 45.34% after FFF attack), as reported in Table 5.1. The mAP for Faster RCNN in the case of FFF perturbations is 45.34%, whereas when *Relabelling Model* is applied, it increases performance to 49.39%. Similarity in terms of UAP, the mAP for Faster RCNN is 46.40%, and 50.72% for *Relabelling Model*. Clearly, we can notice that FFF attack has more impact on the performance of Faster RCNN and *Relabelling Model* with $\approx$ 1% fooling rate compared to UAP.

Furthermore, looking at the AUC scores, Table 5.2 shows the AUC scores for 14 randomly chosen objects, to compare the scores for Faster RCNN detector and the *Relabelling Model* in terms of both FFF and UAP perturbations. Similarly, the *Relabelling Model* achieves a higher average of AUC scores in both attacks than Faster RCNN, because this contextual model does not only improve scores, but also suggests new labels upon contextual knowledge obtained from the scenes. The reader is referred to Appendix E.1, for all objects AUC scores.

Observing from the statistical results shown in Tables 5.1 and 5.2 that the *Relabelling Model* can help to improve the detector even when it is attacked. The *Relabelling Model* achieves higher mAP and F1 scores compared to the detector when both types of attack are applied, which is expected from such an experiment. This is because the attack affects the detector, which depends on objects appearances, to misdetect/misclassify some objects. However, the *Relabelling Model*, which depends on contextual information, can help improving performance and minimising the issue of misclassification due to the inclusion of context.

In terms of visual results, Table 5.3 shows the results for some images after both FFF and UAP perturbations are added. For each result illustrated, the original inputted image (before perturbation added) and perturbed image are presented (left and right, respectively), followed by the outputs from both Faster RCNN and the contextual *Relabelling Model*.

First, the FFF perturbations are added on image as results show. At the first glance, it might be so hard to distinguish between the original image and the perturbed image, and thus, they have been labelled to avoid any confusion. It can be seen while there are so many pre-defined objects in the image, which the detector is trained to recognise (*i.e.*, `buses`), it only detects two objects. One is correctly detected, which is a `bus` with confidence of *0.9523*, and the other incorrectly detected, which is a `boat` with confidence of *0.7772*. This is due to the perturbation added. The *Relabelling model* is then applied, which corrects the `boat` to a `bus`, but with confidence of *0.5676*. Therefore, it can be seen how important contextual information even when images are perturbed. Yes, it can be said that the *Relabelling Model* just changes one label, but as previously stated, it is a post-processing step, thus it only judges detector outputs.

Moreover, the *Relabelling Model* does not only change labels, but only remove them in case no alternatives are found to suit the observed contextual information obtained from the image. As presented in the second image, a `motorcycle` is detected. Faster RCNN detects the object correctly, but incorrectly detects other parts of it as a `suitcase` and a `person`. Due to the size and location of the incorrectly-detected objects with respect to the `motorcycle`, the *Relabelling Model* removes them, and decreases the confidence of the `motorcycle`. As stated earlier in Chapter 4, the *Rescoring Model* works better when there are several objects presented, as it can builds the decision upon their contextual information. However, in this case, because the *Relabelling Model* removes other objects, I believe this is the reason of reducing `motorcycle` confidence.

Finally, in the last shown result for FFF, we can see that the *Relabelling Model* performs slightly poor. First, Faster RCNN detects four objects: two incorrect, which are a `bottle` and a `pizza`, and two correct, which are a `sandwich` and a `dining table`. The *Relabelling Model* removes the `pizza` and labels as a background, which is a good result. However, it replaces the `sandwich` to a `cake`, making it incorrect detection, while it was initially correct. Such a result shows a good performance of the model, but also poor one by replacing the correct object into incorrect. This can be because the detector is not very confident with the `sandwich` (*i.e.*, confidence is *0.7041*), and thus the *Relabelling Model* suggests the `cake`.

From the three illustrated results, we can see that the detector misclassifies some objects. However, the *Relabelling Model* corrects labels and increases scores, which can be said that this model helps reducing the impact of the attack.

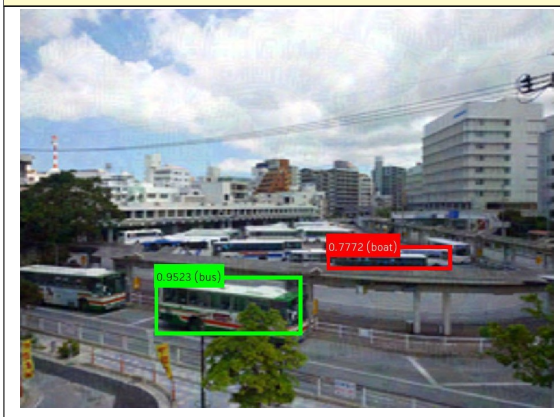
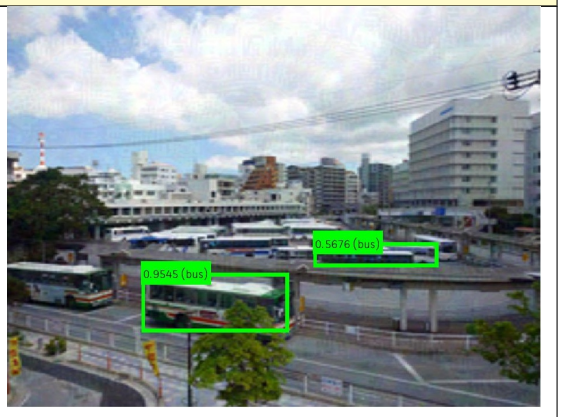
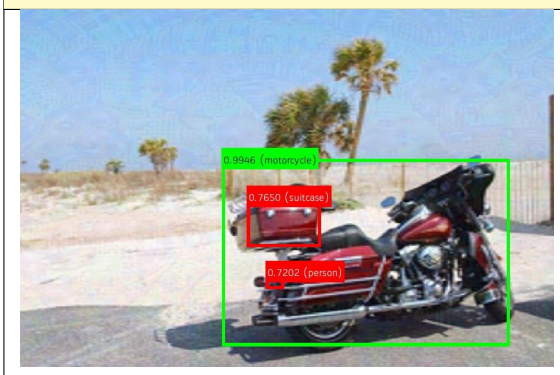
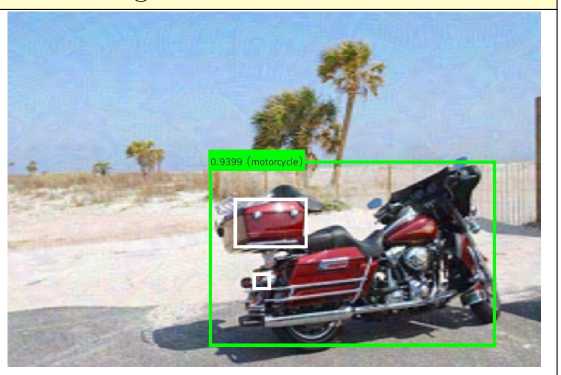
In terms of UAP perturbation when added to image, as presented in the second half of Table 5.3. The first image shows several `cows`. The detector detects one of the `cows` as a `sheep`, whereas when inputted into the *Relabelling Model*, it has been correctly to be a `cow`, as that can seem more likely due to the presence of other `cows` in the scene. Similarly, as shown in the following image. The *Relabelling Model* contributes to correct the label from a `cat` to a `dog`. Even though, the detector has detected the `cat` with high confidence (0.7801), the *Relabelling Model* is still able to correct the object with a confidence of *0.7488*. This can be due to the presence of the other detected object, which is a `fire hydrant` indicating an out-door scene. Statistically, looking at the co-occurrence matrix, as presented in Appendix A, the likelihood of a `dog` co-occurs with a `fire hydrant` is higher than with a `cat`.

In the third image, three objects are detected: two `toilets`, and a `chair`, where the `chair` is incorrectly detected. The presence of a `chair` is a unlikely in such a position, thus the *Relabelling Model* removes it, where no replacement is suggested. This can be because either the detector did not suggest the `sink` in the first top possibilities, or the *Rescoring Model* scores all possibilities with less than the threshold set (*i.e.*, 0.4).

Table 5.3.: Results: Faster RCNN and Relabelling Model outputs for FFF and UAP perturbed images: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively

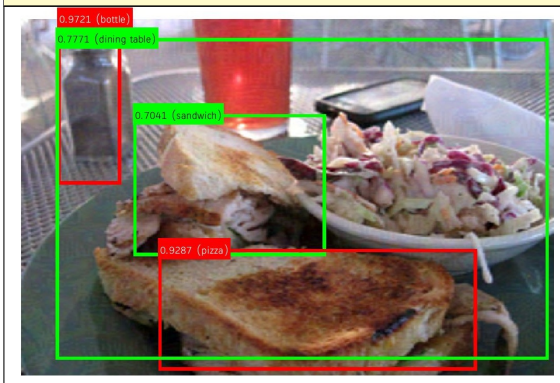
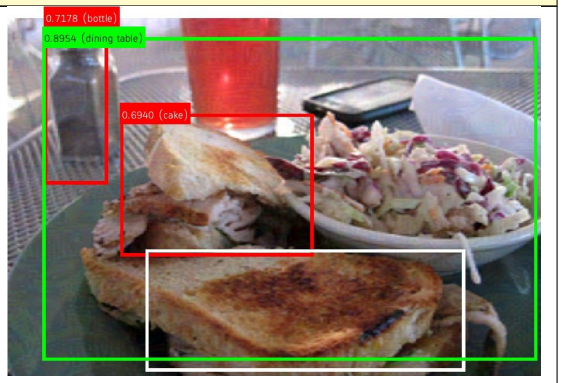
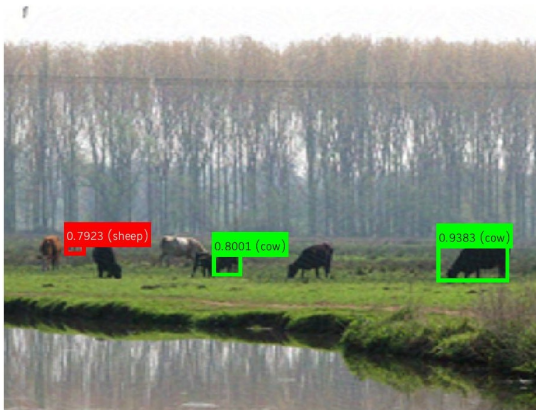
FFF Perturbation	
Original Image	Perturbed Image
	
Continued on next page	

Table 5.3 – continued from previous page

Faster RCNN	Relabelling Model
	
Original Image	Perturbed Image
	
Faster RCNN	Relabelling Model
	
Original Image	Perturbed Image
	

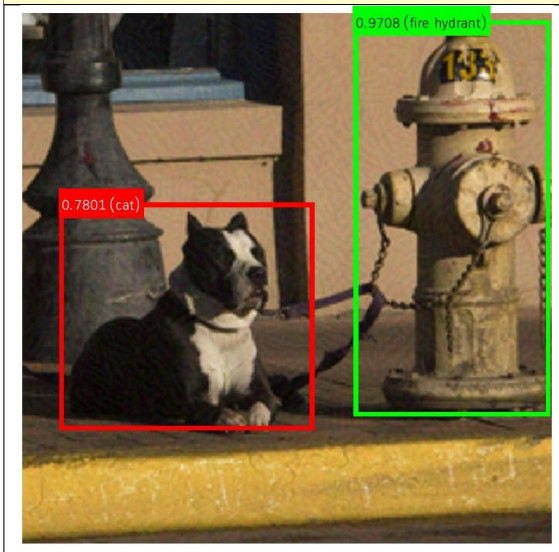
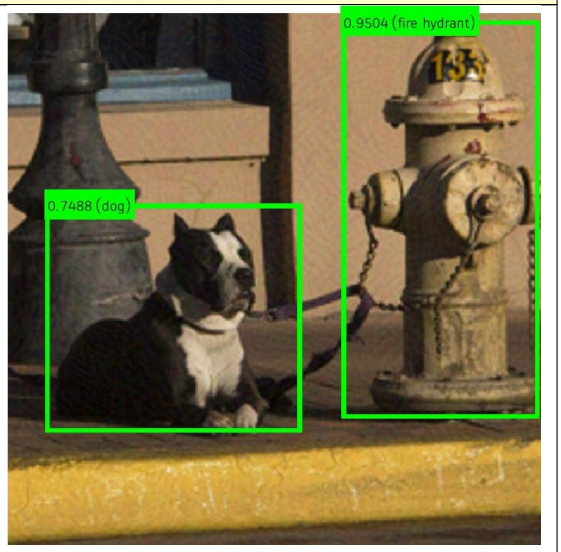
Continued on next page

Table 5.3 – continued from previous page

Faster RCNN	Relabelling Model
	
UAP Perturbation	
Original Image	Perturbed Image
	
Faster RCNN	Relabelling Model
	

Continued on next page

Table 5.3 – continued from previous page

Original Image	Perturbed Image
	
Faster RCNN	Relabelling Model
	
Original Image	Perturbed Image
	

Continued on next page

Table 5.3 – continued from previous page

Faster RCNN	Relabelling Model
	

To sum up, when the perturbations are added to the entire image, a drop of almost 17% is shown in the detector performance, which results to a decrease in the *Relabelling Model* competence. In comparison between the application of FFF and UAP, FFF seems to impact images more than what UAP does. As presented in the results shown in Tables 5.3, 5.2 and Table 5.1 that detector detects less objects when FFF is added to inputted image. In both cases where FFF or UAP is applied, the *Relabelling Model* shows an improvement in enhancing the detector performance, and correct some of the detector outputs when it fails to correctly detect objects.

Because perturbations are added to images, which are passed into single-object models, as presented in the studies (Section 5.2), in the following Section (5.3.2), perturbations will be added to each region of interest (RoI) that the detector detects during the detection process, unlike the process followed in this section, as perturbations are added to the entire image before passed into the detector. It is expected that applying perturbations on RoIs would decrease the performance of the detector even more, and that would help us investigating the performance of the *Relabelling Model* as well.

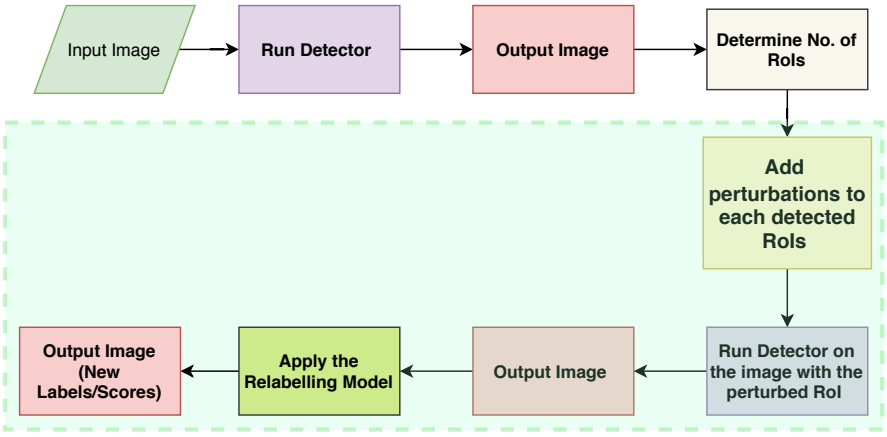


Figure 5.6.: Apply perturbations per detected Region: Green box presents steps that are repeated as per number of detected object simultaneously

5.3.2. Experiment Two: Perturbation On RoIs

As shown in experiment one, above, that the perturbations added on images noticeably drop the performance of the detector. In this experiment, I am examining the impact of the attack not on the entire image, but rather on each region of interest (RoI) to test the performance of detector and the *Relabelling Model*. Steps followed in this experiment are slightly different from the ones stated in experiment one, as presented below.

As shown in Figure 5.6, I am following the steps below to examine the impact of contextual information on adversarial perturbations on each region detected.

1. Input an image from MSCOCO 2017.
2. Run Faster RCNN.
3. Obtain Faster RCNN outputs.
4. Determine the number of the detected objects.
5. Add perturbations to each detected region.
6. Run the Faster RCNN, again, on the image with the perturbed regions.
7. Obtain Faster RCNN outputs (new scores/labels only for the perturbed regions).
8. Apply the contextual *Relabelling Model*.
   NOTE: Steps 5-8 are repeated upon the number of the detected regions, where the scores/labels are only chosen after regions are perturbed. These steps are presented in the green box as in Figure 5.6.

Impact of Perturbations on Detected Regions

As the perturbations attack examined in this experiment is initially developed, and tested on single objects, thus the attack, here, will be applied only on single objects (*i.e.*, detected regions), where all detected objects are to be perturbed, but individually.

In regards to Faster RCNN performance before and after attack is added on regions, is illustrated in Figure 5.7 and 5.8. Please note that the illustrations are presented in two figures to provide visualisation simplicity as possible. In the first figure, the original image and Faster RCNN predictions before perturbations added are shown in the top row (left and right sides respectively). We can see that Faster RCNN detects five objects, which are two `persons`, a `cup` (hidden behind the dining table label), a `cake` and a `dining table`. As the steps for this experiment are followed, the perturbations are to be applied on each detected region individually. Starting with Region One (*i.e.*, `person`). It contains a boy detected with confidence of *0.9988*. Therefore, the region attacked, as shown in the second row, where the patch is presented before the attack (left) and after attack (right). The image with the perturbed region is passed into the detector again as shown in the last row, where outputs of Faster RCNN is presented on the left, and the *Relabelling Model* on the right. It is noticed that the two `persons` initially detected, are misdetected. This is because both objects are located within the perturbed region. More importantly, even a part of the `dining table` is attacked, but the detector still detects it with a higher confidence compared to its confidence before the attack.

Moreover, in Figure 5.8 (*i.e.*, the second figure), some other regions are attacked. In the first row, the `cake` region is cropped and presented before and after the attack. As illustrated in the following row, the detector detects the object correctly, but with a slightly lower confidence (*i.e.*, *0.9762* before attack, and *0.9602* after attack). However, the *Relabelling Model* increases the confidence to *0.9860*. We notice that even when this region is attacked, the detector still manages to detect the object, which I believe is due to its size which is to be discussed below in more details. Finally, for the last detected region, which is the `dining table`, we see that the detector detects it incorrectly as a `couch` with confidence of *0.7675* after perturbations added. However, when the *Relabelling Model* is applied, it corrects it to `dining table` with *0.8167* confidence. Such results show how the perturbations added, even to regions, can impact the decision of the detector. Because the *Relabelling Model* cannot predict undetected objects as it is dependent on the outputs of the detector, the performance of detector is illustrated here. In the remaining of this experiment, only the detector predictions after attack are illustrated.

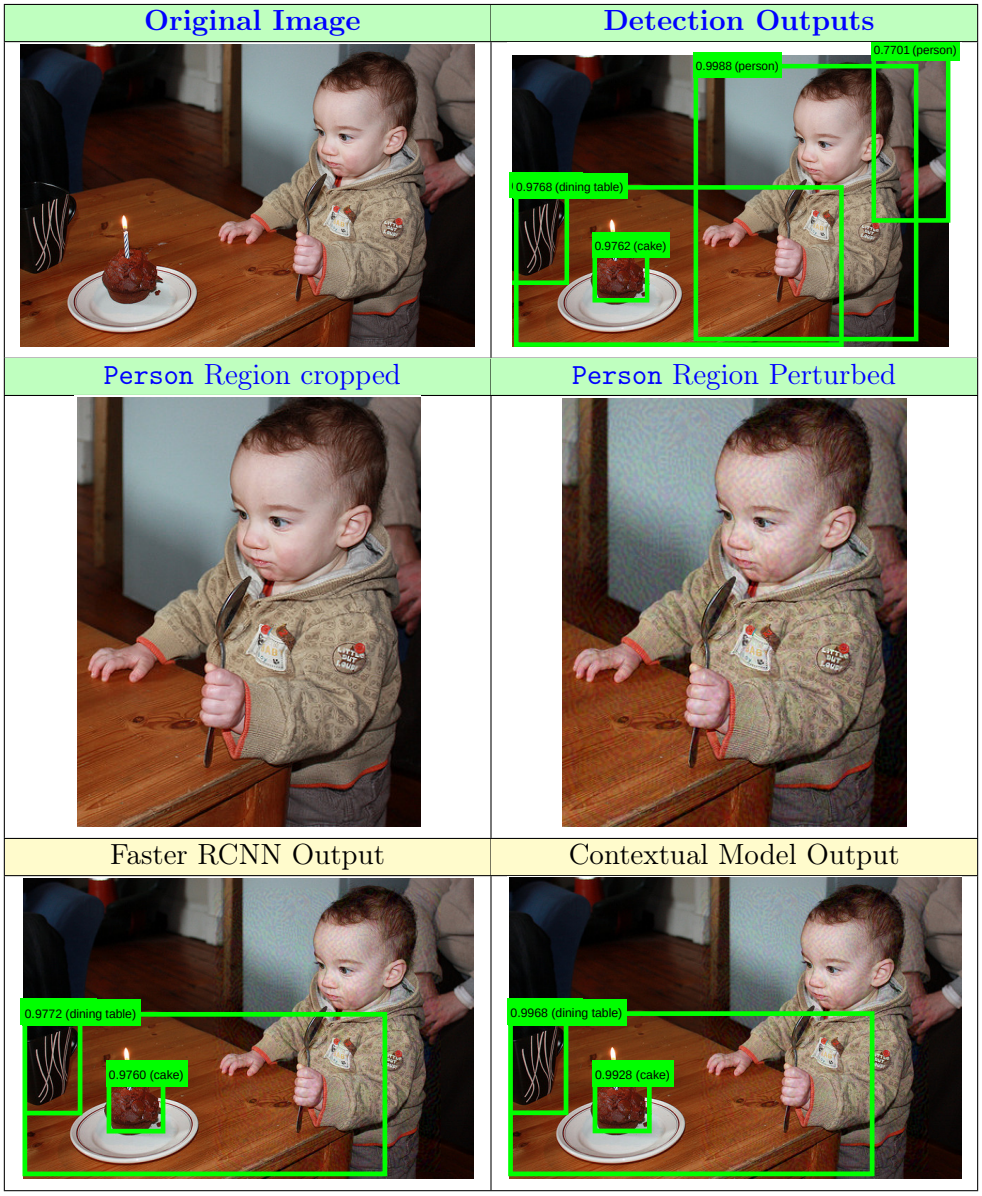


Figure 5.7.: Comparison between Faster RCNN and the Relabelling Model before and after perturbations applied.

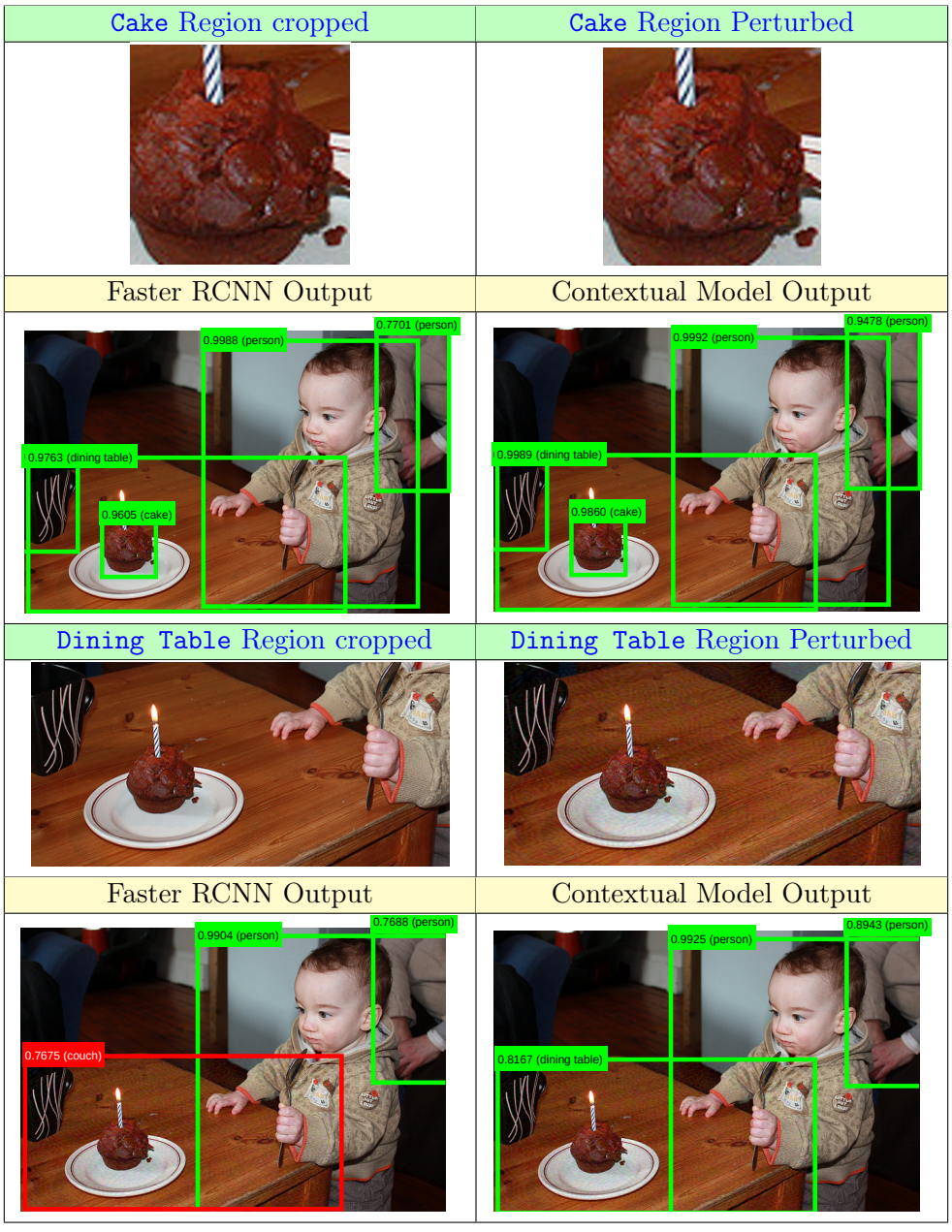


Figure 5.8.: Continued: Comparison between Faster RCNN and the Relabelling Model before and after perturbations applied.

Table 5.4.: mAP and F1 scores in percentages [%] for the Faster RCNN and the Relabelling Model on perturbed regions.

	FFF		UAP	
Models	$mAP_{0.5}$	F1 Score	$mAP_{0.5}$	F1 Score
Faster RCNN	26.25%	20.78%	23.18%	18.17%
Relabelling Model	**29.05%**	**23.14%**	**26.36%**	**20.53%**

Performance of Models On Perturbed Regions

As shown above, Faster RCNN fails to detect some objects when the attack is applied on regions. Statistical results showing the performance of Faster RCNN and *Relabelling Model* after the attack is added on regions is presented in Tables 5.4, and 5.5.

First, Table 5.4 shows the mAP scores for both the detector and the contextual model where both types of attacks are applied. In terms of FFF, the performance of Faster RCNN is dropped by ≈ 40 points compared to its performance before the attack, and almost half compared to when attack in applied on the entire image. Similarly, for the contextual model in both cases of attack. This can lead us to believe that when perturbations are applied on regions, they have more impact that when applied on the entire image. This could be because such perturbations are more effective when applied on small regions and single objects. As I presented above, in Figure 5.7, the `cake` is detectable even it is attacked due to its small size. More results and examples are explored and discussed later.

Moreover, it is observed that UAP perturbations when applied on regions has more impact on models performances compared to FFF. This is in contrast of when perturbations applied one the entire image, as FFF has impacted models performance more, as previously reported in Table 5.2. This can also be due to the training done for UAP, as I have cropped all regions detected (before perturbation added) and train the UAP model on, thus its used to small regions and not the case as when applied on the entire image.

In terms of the AUC scores comparison, Table 5.5 shows a comparison between some objects in terms of Faster RCNN and *Relabelling Model* on perturbed regions when both FFF and UAP attacks are applied. The average AUC scores for both Faster RCNN and the *Relabelling Model* are significantly dropped, but is lower in the case of UAP attack. In comparison of both models we can see that the *Relabelling Model* shows a better performance in the majority of classes AUC scores with an increase of almost 0.04 points. The reader is referred to Appendix E.2, for all object classes AUC scores.

As reported above, when the perturbations added to the detected regions, they are then detected again, thus the number of detected objects might be considerably dropped. Table 5.6 shows some results how attacks impact the detection outputs. Both FFF and UAP perturbations are added and the results are illustrated. The original inputted images, the detected regions before attack, and outputs of both Faster RCNN and the *Relabelling Model* are presented, for each examined image.

In the first section of the table, FFF perturbation when added to detected regions is shown.

Table 5.5.: A comparison between some MSCOCO object classes AUC scores between detector and Relabelling Model on perturbed regions.

	FFF		UAP	
Class Label	Faster RCNN	Relabelling Model	Faster RCNN	Relabelling Model
Person	0.18573	**0.26869**	0.14653	**0.23466**
Motorcycle	0.11676	**0.14451**	0.12056	**0.12799**
Train	0.05501	**0.22815**	0.03883	**0.51941**
Cat	0.06807	**0.22261**	0.03990	**0.19880**
Horse	**0.08057**	0.06162	**0.13774**	0.09259
Giraffe	**0.10683**	0.08974	**0.12179**	0.08974
Sports Ball	0.46581	**0.56902**	0.33207	**0.36737**
Bottle	0.20898	**0.27789**	0.15695	**0.24819**
Wine Glass	0.27083	**0.36938**	0.24791	**0.25510**
Donut	0.14443	**0.19957**	0.04464	**0.04761**
Bed	0.01515	**0.01538**	0	0
TV	0.09269	**0.16998**	**0.09116**	0.08974
Microwave	**0.17241**	0.16071	0.06896	**0.10714**
Vase	0.24561	**0.3**	0.20116	**0.24611**
Mean	0.14310	**0.19470**	0.12043	**0.16994**

In detail, as shown in the first image, the original image is presented on the left, and the detected regions hightailed are presented on the right. Regions before the attack are shown, regardless whether they are correct or incorrect, to illustrate how the number of detected regions can be affected/dropped by the attack. For example, Faster RCNN detects five objects before the FFF attack is applied. However, when attack applied per region, Faster RCNN only detects two objects, which are a `person` and a `motorcycle`. More than half of the objects are undetected when attack applied. The *Relabelling Model* is then applied, which helps to increase scores for both objects.

In the following image, we can see that detected regions after attack is not too much impacted, as the number is not considerably different. Ten objects are detected before the attack, whereas seven after the attack. The *Relabelling Mode* seems to improve results as one of the incorrect detected objects is relabelled as background. In this result, as noticed, only three regions impacted by the perturbation, where all others are sill detectable. I believe this is because of their sizes. In other words, it is observed that the larger the region is, the more it is impacted by the attack, the harder for the detector to predict.

Similarly, in the last image, three objects are detected before the attack, which are two `cups` and a `pizza`. When the attack is added, only one object detected incorrectly, which is a `bottle`. Applying the contextual model helps to correct it to a `cup`. Noticeably, the object `pizza` is misdetected, which I believe is due to its size.

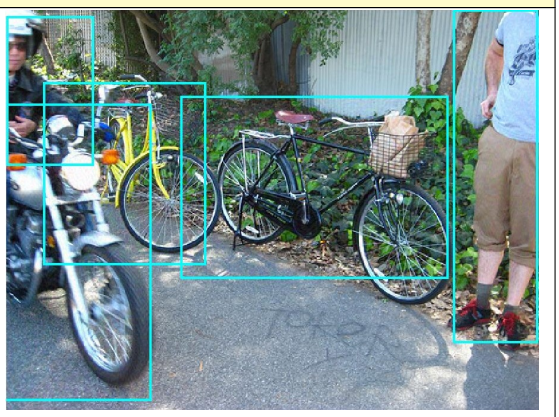
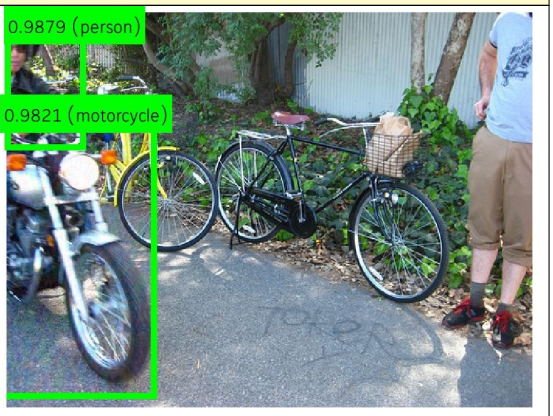
In terms of the application of UAP perturbation, the same image used in FFF attack is inputted here to illustrate differences. We can see in this type of attack, only one object is detected after the attacked, where as in the case of FFF, two are detected. The *Relabelling Model* is applied, which reduces the score of the object. I believe this is because only one object is detected, and thus the model lacks the contextual information among objects. More importantly, we can see that UAP leads the detector to misdetect more regions

compare to FFF, which is in line with the statistical results we obtained as UAP has more impact the FFF in regions.

In the following image, the number of detected regions before attack are five, but after attack, they are reduced to three, which are two `persons`, where one is incorrectly detected, and a `handbag`. The *Relabelling Model* is then applied, which corrects the incorrect detection to a `handbag` with a high confidence *0.9783* due to its co-occurrence with other objects.

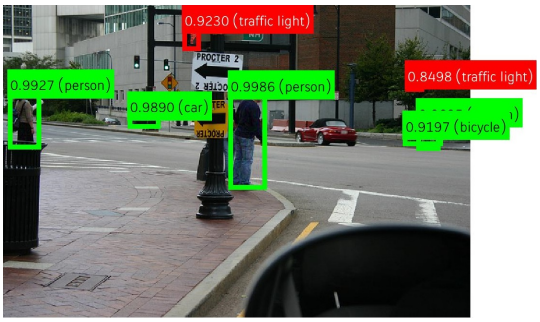
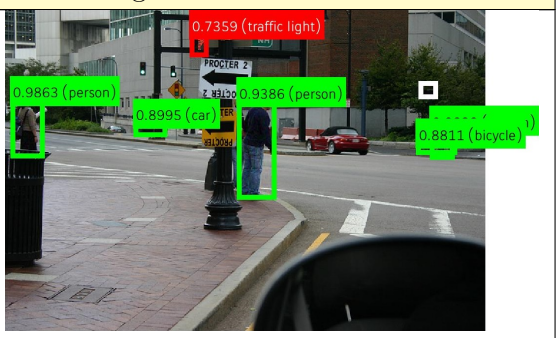
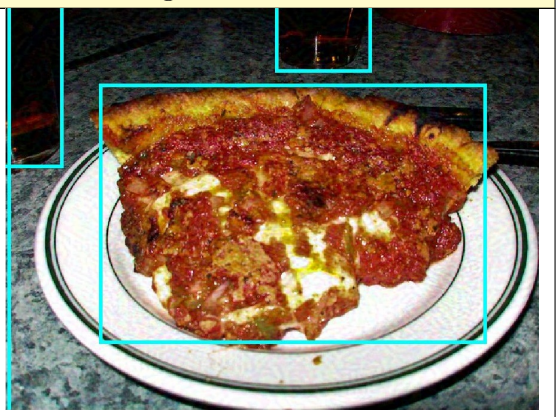
In the last row images, it can be seen that five objects are detected, but later when the attack is added, only one object is detected (*i.e.*, a `cow`), but incorrectly. As stated earlier, the *Relabelling Model* is not the best tool to correct detection is such cases, due to the presence of only one object. It relabels the incorrect detected object into a `dog`, which is still incorrect.

Table 5.6.: Results: Faster RCNN and Relabelling Model outputs for FFF and UAP perturbed regions: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively.

FFF Perturbation	
Original Image	Detected Regions Before Attack
	
Faster RCNN	Relabelling Model
	

Continued on next page

Table 5.6 – continued from previous page

Original Image	Detected Regions Before Attack
	
Faster RCNN	Relabelling Model
	
Original Image	Detected Regions Before Attack
	
Faster RCNN	Relabelling Model
	

Continued on next page

Table 5.6 – continued from previous page

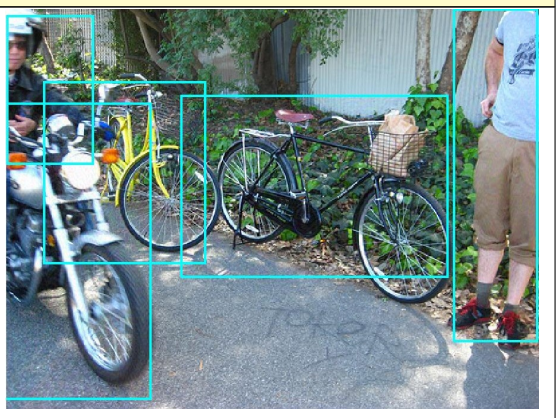
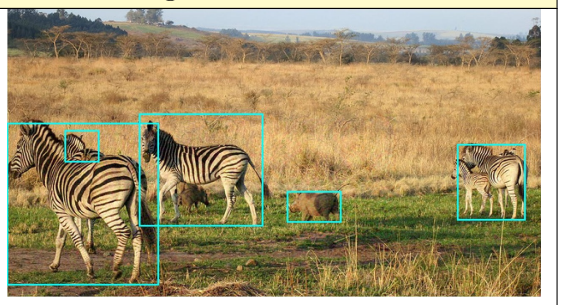
UAP Perturbation	
Original Image	Detected Regions Before Attack
	
Faster RCNN	Relabelling Model
	
Original Image	Detected Regions Before Attack
	
Continued on next page	

Table 5.6 – continued from previous page

Faster RCNN	Relabelling Model
	
Original Image	Detected Regions Before Attack
	
Faster RCNN	Relabelling Model
	

In short, I claim, upon the statistical and visualised results obtained in the above two experiments, that perturbations do negatively impact the performance of the detector leading to a drop in the performance of the contextual model. In comparison between the two experiments, as reported, attack on regions drops the chosen detector performance considerably compared to the case where attack is applied on the entire image. It is also noticed that the larger regions, the more the detector fails to detect.

5.4. Summary

This chapter presented the definition of perturbations, which are some computed function added to image leading to a drop in models performance. Perturbations can be generated on X dataset for X model, but they can still impact Y model that is even trained on Y dataset, as the case of FFF adversarial examples. Two different types of attacks are used, which are UAP and FFF. The former is claimed to impact more when trained on the same dataset used in testing, thus it is data-dependent. However, the latter is data-independent, and used as proposed and trained on ImageNet, but examined on MSCOCO 2017.

The performance of Faster RCNN and the contextual model drop significantly when attacks are applied compared to their performances on natural images. However, the contextual model still achieves better results than Faster RCNN. This is expected, as this model solely relies on the contextual knowledge observed from the digital scene. It, therefore, can be seen an tool to mitigate the impact of adversarial perturbations, where some of the advantages of this tool, compared to the methods discussed in Section 5.2.4, are as follows:

1. The contextual model is applied as a post-processing step, meaning that there is no need for modification in the model, and data pre-processing. This is in contrast to Akhtar et al. (2018), which requires data pre-processing prior to passing them into the target model.

2. Compared to Shafahi et al. (2018), which is seen as time-consuming due to the inclusion for data training, and is limited to the types of attacks applied in training. The contextual model, however, can be applied on any type of attacks as it is dependent on the detector' outputs and not the perturbation types.

3. The contextual model is applied on both groups of detectors (*i.e.*, one-stage and two-stage detectors). However, other methods such as Shafahi et al. (2018); Mummadi et al. (2019) require accessing the feature extraction stage in the model, which thus limit them to only two-stage detectors.

Faster RCNN depends on the visual appearance obtained from regions, whereas the *Relabelling Model* depends of the contextual statistics, which are explicitly extracted. Developing a model that can benefit from both models, and extract contextual features implicitly might it be a solution. Such a model is proposed in Chapter 6, below. It is expected to perform higher than Faster RCNN and the *Relabelling Model* as it takes advantages from both models and then combined into one.

Chapter 6

TRANSFORMER ENCODED FEATURES

6.1. Overview

This chapter proposes a new context module, called *Transformer-Encoder Detector Module*, that can be applied to an object detector to (i) improve the labelling of object instances; and (ii) improve the detector's robustness to adversarial attacks. This module is based on the use of multiple models to implicitly use contextual information presented in the scene. It uses an attention mechanism that enables learning the contextual knowledge. In this chapter, it will be evaluated using both natural and perturbed images in comparison with Faster RCNN and the *Relabelling Model* proposed in Chapter 4.

6.2. Background

Deep Convolutional Neural Network (CNN) (Karpathy and Li, 2014) Recurrent Neural Networks (RNNs) (Cho et al., 2014; Vinyals et al., 2014), and Long Short-Term Memory networks (LSTM) (Soh, 2016) have been widely used, showing great performances in the field of computer vision and sequence modelling tasks such as machine translation and language modelling (Rennie et al., 2016). However, due to their sequential fashion of processing word-by-word (*i.e.*, element-by-element) which suffers from forgetting the content of distant positions in sequence or mix them with following positions' contents. They are also an obstacle for the parallelisation process, which thus leads to heavy training. Attention mechanisms, therefore, are proposed to overcome such issues. Attention is introduced to solve the issue of slow training and the model forgetting, which is the backbone of the Transformer (Vaswani et al., 2017). The Transformer is an important part adapted for the proposed method in this chapter. Vaswani et al. (2017) propose the Transformer architecture for natural language processing, as a way to encode word context from documents corpora using dot product attention between word vectors. It is completely dependent on

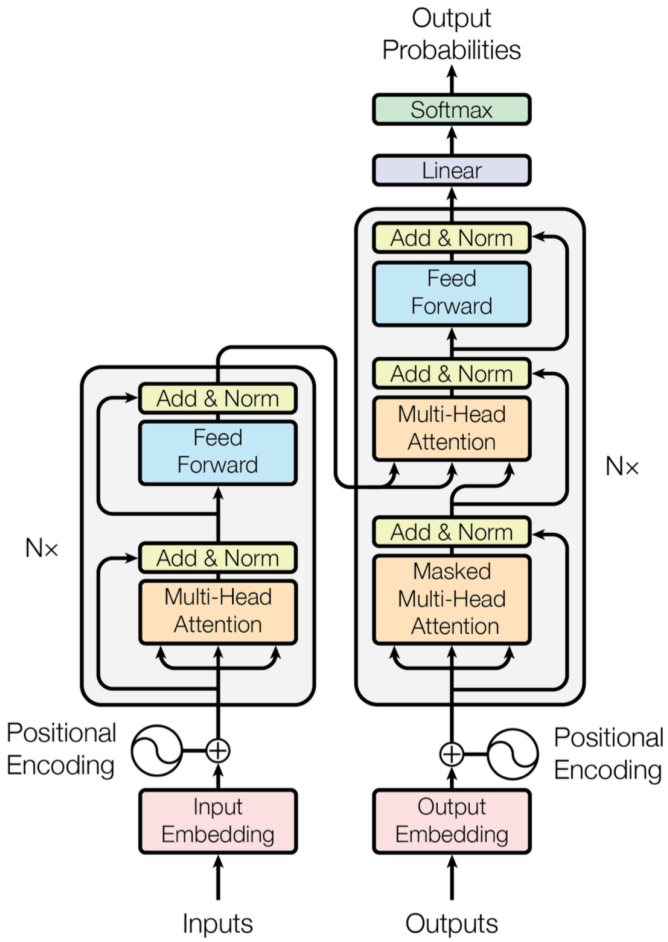


Figure 6.1.: The Transformer Architecture, adapted from Vaswani et al. (2017)

the use of attention, as it links the input and output in a parallel process. It processes the entire sentence (*i.e.*, in the case of NLP) at once, unlike the previously stated methods, which encode the sentence a word-by-word. It, therefore, improves the training time and performance as well. The Transformer consists of a set of encoders and decoders, which are composed of a stack of layers. In terms of encoders: each has two major components, which are (1) self-attention layer and (2) feed-forward neural network. On the other hand, the decoder has two similar components and an additional one: (1) self-attention layer, (2) encoder-decoder layer, and (3) feed-forward neural network. The Transformer architecture is shown in Figure 6.1.

Because the Transformer was first introduced for the purpose of language translation, it is explained in this concept. First, the encoder takes the entire sentence as the input. It is first converted to embeddings (*i.e.*, convert input tokens into vectors). Since there is no recurrence in the model, the positions of the tokens are needed, thus positional encoding is applied, which "*injects some information about the relative or absolute position of the tokens in the sequence*", using sine and cosine functions of different frequencies (Vaswani et al., 2017). In other words, it is used to obtain the context of each word in the sentence.

In this stage, the produced positional information and the embeddings are ready to input into the encoder block. The encoder starts with attention, which analyses which part of the input should be the focus answering how relevant each word in the sentence with

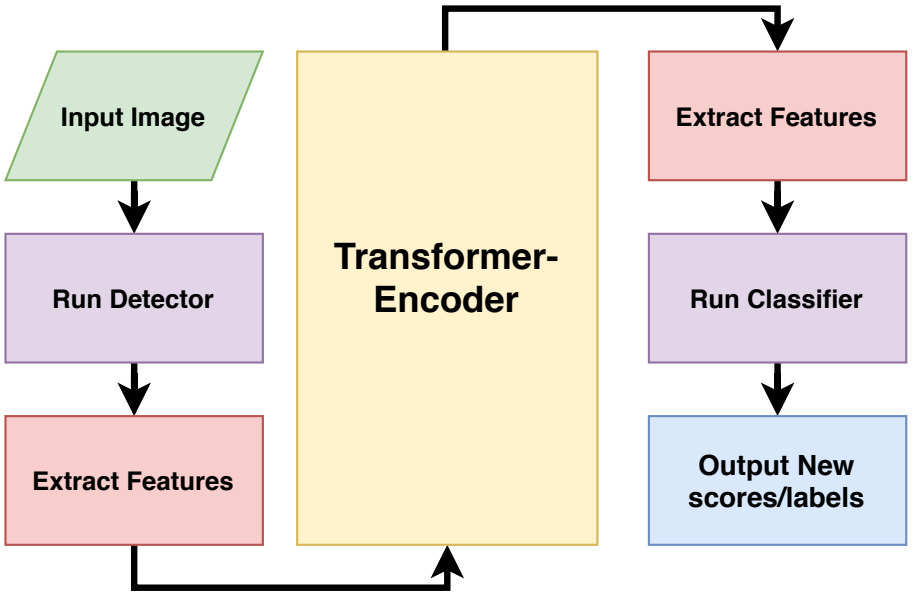


Figure 6.2.: Transformer Encoder-Detector Module Architecture.

others. Each attention vector represents the contextual relationships between words in the sentence inputted. The feed-forward NN is then applied to each attention vector, which is then passed into the decoder.

The decoder does similar processes, but with the other side of the translation (*i.e.*, the target language). It takes the tokens and the computed positional information and passed into the decoder block. Attention representing how each word related (as in the encoder) is applied, followed by an encoder-decoder attention layer. This layer takes the attention vectors (produced by the decoder itself and the vectors passed from the encoder). In this layer is where the actual mapping between the two languages occurs. The vectors produced are then passed individually into the feed-forward NN to be ready for the next step (*i.e.*, the next decoder block or the linear layer). The linear layer is also a feed-forward layer that takes the outputs from the decoder and passes them into the softmax layer. Finally, the softmax converts them into a probability distribution to the next word prediction (Vaswani et al., 2017).

The Transformer architecture, due to its performance and speed, has been used in a variety of approaches in NLP, such as (Devlin et al., 2018; Young et al., 2018; Yang et al., 2019; Radford, 2018), and has recently been applied to computer vision problems (Carion et al., 2020).

6.3. Method

A novel model is proposed in this chapter, which is built upon the success of the Transformer model proposed by Vaswani et al. (2017). However, only the Transformer-Encoder is adapted, as shown in Figure 6.2, where the proposed model, named *Transformer-Encoder Detector Module* (TEDM) architecture is illustrated.

First, an image is inputted into the baseline detector (*i.e.*, Faster RCNN) for feature extraction. After detector detects objects, features representing the regions for the detected objects are extracted. As the dimension of the features extracted is 4069, they are mean-pooled to 2048 to suit further processing. The pooled features and the boundary boxes obtained from the detector representing the detected regions are passed into the Transformer-Encoder with a dimension of 2048. Boundary boxes are included to encode the spatial features for each region and extracting the contextual information among regions implicitly.

The Transformer Model is then applied, which is already pre-trained on MSCOCO 2014, adapted from Krasser (2020); Vaswani et al. (2017). It takes the pooled features and the boundary boxes to process them with the attention mechanism to output features with a dimension of 512. A feed-forward NN classifier with 200 hidden units is then applied (See Section 4.2.3 in Chapter 4, for more information about the used classifier). The classifier produces the new scores and predicted labels for each region.

This method includes the Transformer-Encoder to benefit from the use of attention and the positional encoding operation occurred before features are passed into the encoder. The operation, as said, helps to simply encode the semantic and spatial contexts for each region inputted implicitly. Due to the time limitations, an end-to-end model could not be developed, thus the *Transformer-Encoder Detector Module* (TEDM) is proposed to take advantages from the detector in extracting features, and from the encoder in producing small features in the dimension that includes contextual information implicitly.

TEDM is examined, as below, on MSCOCO 2017 validation dataset in comparison with Faster RCNN (*i.e.*, baseline detector) on two types of images: (i) Natural images, where no perturbations are added; and (ii) perturbed images.

6.4. Experiments

This section presents the results obtained from the application of the proposed method (*TEDM*). Images before any attacks, called natural images, are used to examine the impact of this model in comparison with Faster RCNN, as in Experiment One (Section 6.4.1). Experiment Two (Section 6.4.2) illustrates the examination of the *TEDM* vs Faster RCNN on perturbed images. *Please note* that the same images used previously in this thesis are used in those experiments, which are taken from MSCOCO 2017 validation dataset. Only the images with more than one object detected by Faster RCNN are used to compare the performance with the *Relabelling Model* proposed in Chapter 4; thus, consistency is assured.

6.4.1. Experiment One: Natural Images

In this section, I am presenting how effective the *TEDM* compared with Faster RCNN when applied on natural images.

First, three chosen images with one object presented are tested to examine the *TEDM* when only one object is presented. This is because *TEDM* does not solely rely on contextual information as the *Relabelling Model*, but also on the appearance features extracted by Faster RCNN. Therefore, it is expected to work well even with one object presented as illustrated in Figure 6.3, where the predictions obtained from Faster RCNN and *TEDM* are shown on the left and right sides, respectively.

As we can see in the first row, a `person` is detected by Faster RCNN with confidence of *0.9985* is also predicted by *TEDM*, but with very slightly lower confidence as *0.9883*. However, in the second row, a `kite` is detected by Faster RCNN as the only object presented. Applying *TEDM*, in this case, increases the confidence from *0.9647* to *0.9920*. Finally, in the last row, we notice that a `bear` is detected, and passed into *TEDM*, which only reduces the confidence slightly, but still very high confidence. Three images are shown as each presents one object that belongs to a different category. In all three cases, *TEDM* is believed to work very well when a single object is presented. It can be said that *TEDM* is a compatible method producing comparable results as Faster RCNN. Below, more visualised and statistical results are reported where more than one object in the images tested are presented.

Images with more than one object are used to keep consistent with all experiments presented previously in the thesis, and to report comparable statistical results. In Table 6.1, the AUC scores for some objects and all objects average score for the *TEDM* and Faster RCNN are presented. It can be noticed that the *TEDM* outperforms the detector in all of the cases shown obtaining higher average as *0.8922*. This is even better than the *Relabelling Model*, whose mean AUC scores is *0.78278*, as in Table 4.21 (Chapter 4). However, *TEDM* still has lower average compared with *IRM* (*i.e.*, AUC is *0.95314*, as in Table 4.23). This is because the *IRM* does iterate the process of classification 12 times. The reader is referred to Appendix F.1 for AUC scores of the all objects AUC obtained from *TEDM*.

Furthermore, Table 6.2 shows the $\text{mAP}_{0.5}$, mAP (IoU=[0.5:0.05:0.95]) and F1 scores for *TEDM* and Faster RCNN. As presented, the *TEDM* outperforms the detector, with 0.7 points higher performance. Compared with the *Relabelling Model*, *TEDM* still achieves higher mAP and F1. Refer to Table 4.19, for more information about the *Relabelling Model*.

Three images taken from MSCOCO 2017 validation dataset are used to illustrate the performance of the *TEDM* compared with Faster RCNN, as in Table 6.3, where the outputs of Faster RCNN and *TEDM* are shown. Only the outputs of both Faster RCNN and *TEDM* are illustrated for simplicity. In the first row, we can see the predictions obtained from Faster RCNN and the *TEDM* on the left and right sides, respectively. Two objects are detected by Faster RCNN, which are a `bench` and a `dog`. Therefore, the features of those two regions plus their boundary boxes are passed into the *TEDM*, which classifies regions as predicted by Faster RCNN. The confidence of `bench` is increased by the *TEDM*, whereas it slightly reduces the `dog` confidence. Both objects confidences obtained by Faster RCNN and *TEDM* are higher than 0.97, which mean both models are very confident that their predictions are correct.

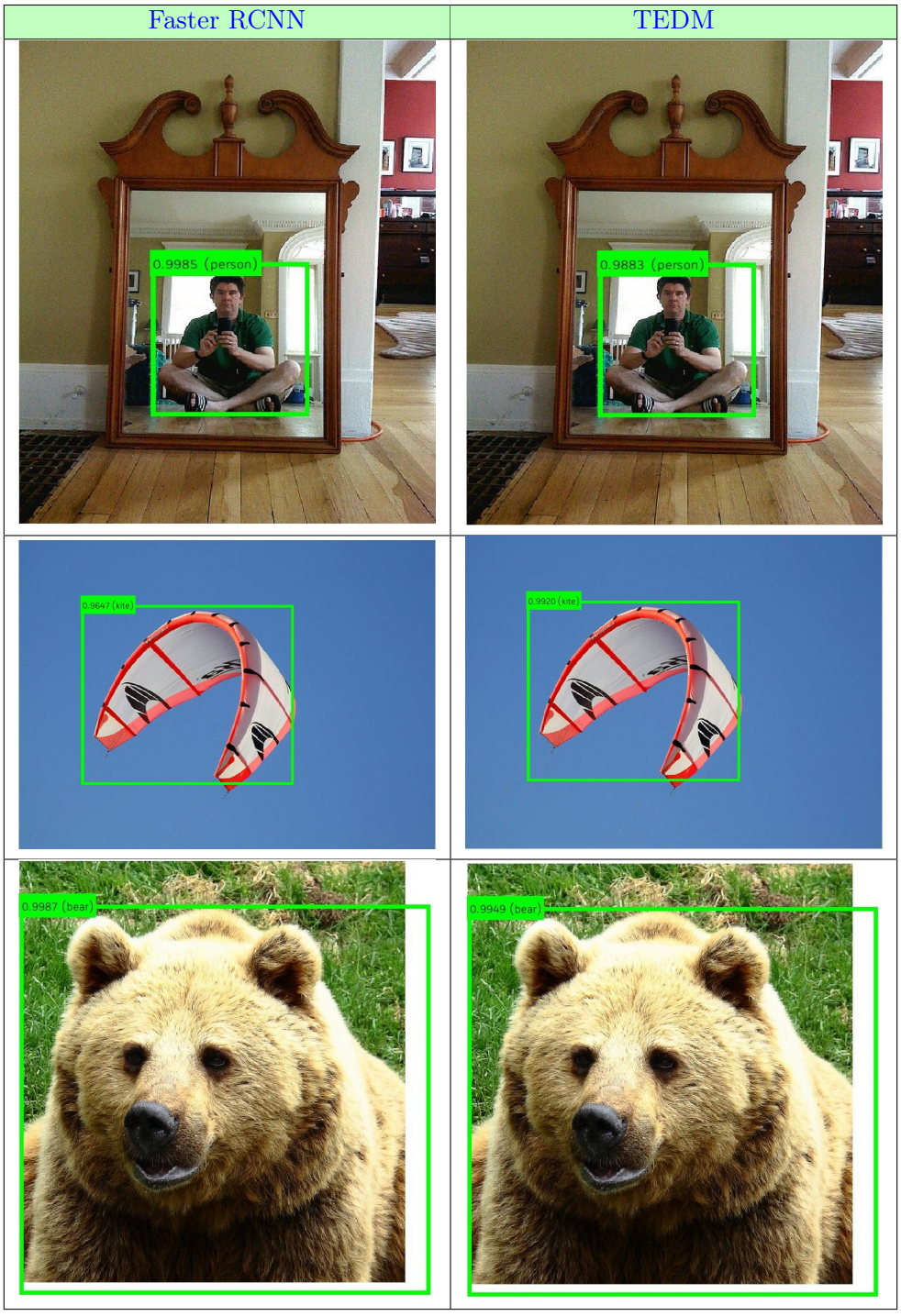


Figure 6.3.: Results: Faster RCNN vs. TEDM outputs on images with a single Object: Green boxes represent correct detections.

Table 6.1.: A comparison between some MSCOCO object classes AUC scores for Faster RCNN and Transformer-Encoder Detector Module.

Class Name	Class ID	Faster RCNN	Transformer-Encoder Detector Module
Person	1	0.84345	**0.95174**
Bicycle	2	0.76073	**0.91346**
Car	3	0.81246	**0.93287**
:	:	:	:
Boat	9	0.699422	**0.88625**
Traffic light	10	0.78979	**0.96175**
Fire hydrant	11	0.94070	**0.99292**
:	:	:	:
Sheep	19	0.78563	**0.92731**
Cow	20	0.82301	**0.93246**
Elephant	21	0.74058	**0.96729**
:	:	:	:
Skis	31	0.72606	**0.87727**
Snowboard	32	0.74229	**0.88235**
Sport ball	33	0.84244	**0.95008**
:	:	:	:
Bowl	46	0.71389	**0.85709**
Banana	47	0.68585	**0.83006**
Apple	48	0.67691	**0.80024**
Mean	-	0.76472	**0.89222**

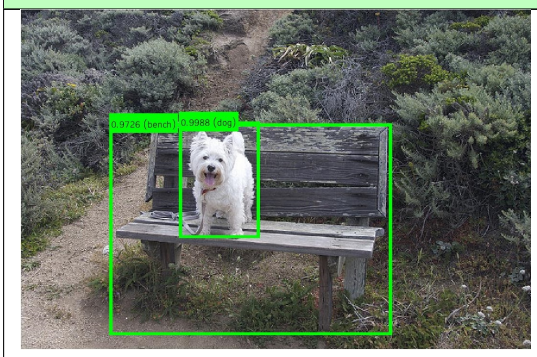
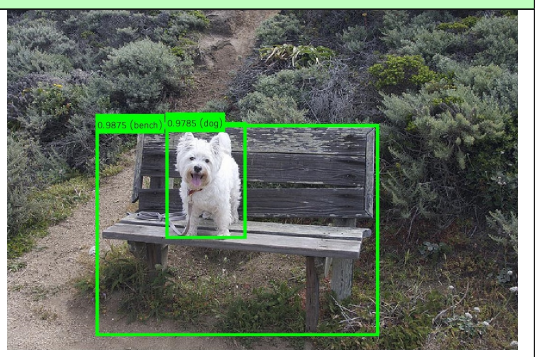
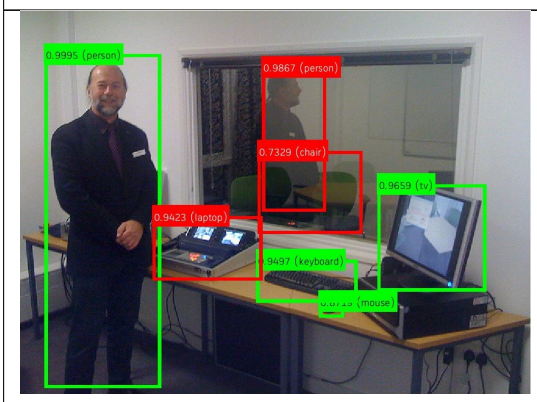
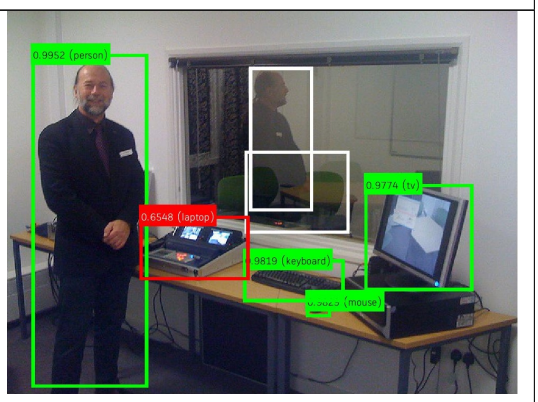
Table 6.2.: mAP and F1 scores in percentages [%] for the Faster RCNN and Transformer-Encoder Detector Module.

Model	$mAP_{0.5}$	mAP	F1
Faster RCNN	62.82	33.48	57.34
Transformer-Encoder Detector Module	**69.07**	**38.19**	**63.27**

In the second row, seven objects are detected by Faster RCNN. Four objects, which are a `person`, a `TV`, a `keyboard` and a `mouse` are detected correctly. However, the other three, which are a `person`, a `laptop` and a `chair` are incorrectly detected. *TEDM* is applied, which removes two of the incorrectly detected objects, but still predicts the `laptop` incorrectly with very low confidence. This result is promising as it shows how the *TEDM* removes incorrect objects and reduces the confidence of the `laptop`. More importantly, in the third image, the *TEDM* outperforms Faster RCNN predictions. It corrects the `bed`, which is incorrectly detected, to a `couch` as correct detection. This clearly shows how the proposed model does not only contribute to removing false detections but also how it rescores and relabels them.

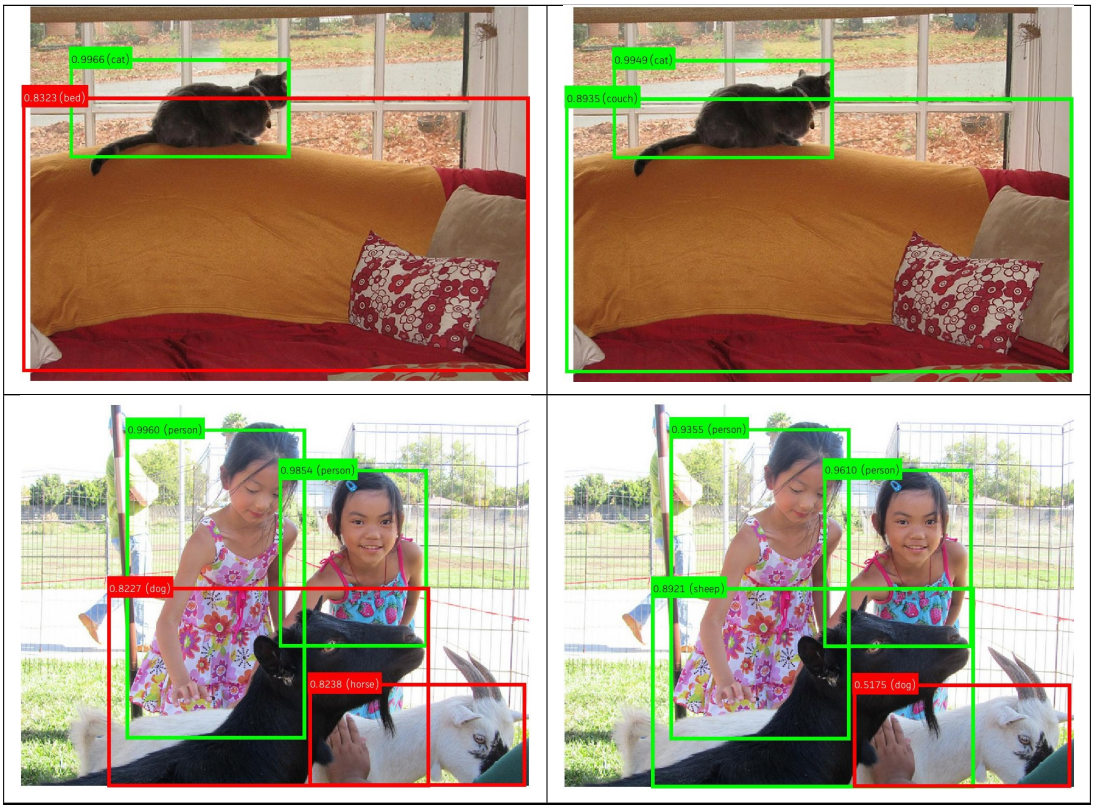
Upon the statistical and visualised results, *TEDM* achieves a better performance than the baseline detector benefiting from the self-attention mechanism. It also helps to correct incorrectly-detected objects. However, it still makes mistakes as presented in Table 6.3 in the last row. Faster RCNN fails to detect the presented two `sheep`, as it predicts them as a `dog` and a `horse`. *TEDM* corrects one them and labels as a `sheep`, but still fails to get the second one, which it labels as a `dog`. Overall, *TEDM* is noticed to do a great job in detecting either single-object or multiple objects, which are better in comparison with Faster RCNN and the *Relabelling Model*.

Table 6.3.: Results: Faster RCNN vs TEDM outputs: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively.

Faster RCNN	**TEDM**
	
	

Continued on next page

Table 6.3 – continued from previous page



6.4.2. Experiment Two: Perturbed Images

As previously attempted in Chapter 5, perturbed images are used to examine the impact of *TEDM*, where perturbations are added on the entire image and per region, as presented below. As perturbations affect both the visual and contextual knowledge observed from the images, they are applied to examine the impact and the effectiveness of the method proposed. In terms of the visual changes, as perturbations added to the images, features extracted are expected to be different from when they are extracted from natural images due to the additions of perturbations. Contextual information, moreover, can be affected because an object may not be detected, and thus a lack of contextual information occurs among objects.

Perturbation On the Entire Image

As explained in Chapter 5, Section 5.3.1, how perturbations are added into images; in this experiment, I am following the exact procedure, but rather to test the *TEDM*. Both statistical and visualised results are reported to illustrate the performance of *TEDM* in comparison with Faster RCNN.

In terms of the statistical results, as shown in Table 6.4. The average AUC scores for the *TEDM* is noticeably higher than Faster RCNN in both types of attack. For the AUC

Table 6.4.: A comparison of AUC scores among some MSCOCO classes obtained from Faster RCNN and TEDM on perturbed images.

	FFF		UAP	
Class Label	Faster RCNN	TEDM	Faster RCNN	TEDM
Person	0.62389	**0.69576**	0.62881	**0.73342**
Bicycle	0.44476	**0.58900**	0.49582	**0.69388**
Train	0.42961	**0.62051**	0.41062	**0.56773**
Boat	0.24063	**0.35673**	0.22854	**0.35638**
Fire Hydrant	0.61949	**0.69811**	0.53459	**0.67924**
Stop Sign	0.76242	**1**	0.69155	**0.78571**
Cow	0.30345	**0.37447**	0.29946	**0.35102**
Bear	0.24074	**0.35833**	0.15	**0.201388**
Kite	0.15154	**0.2192**	0.14601	**0.24780**
Surfboard	0.39266	**0.52464**	0.44750	**0.59365**
Fork	0.39130	**0.54106**	0.28726	**0.44099**
Pizza	0.67393	**0.87767**	0.63185	**0.81783**
Plotted Plant	0.29679	**0.45424**	0.27156	**0.44075**
Toothbrush	0.4	0.4	**0.2**	0.19787
Mean	0.34714	**0.47683**	0.36050	**0.48713**

Table 6.5.: mAP and F1 scores in percentages [%] for Faster RCNN and TEDM on perturbed images.

	FFF		UAP	
Models	$mAP_{0.5}$	F1 Score	$mAP_{0.5}$	F1 Score
Faster RCNN	45.34%	39.13%	46.40%	40.05%
TEDM	**50.59%**	**43.16%**	**51.67%**	**44.00%**

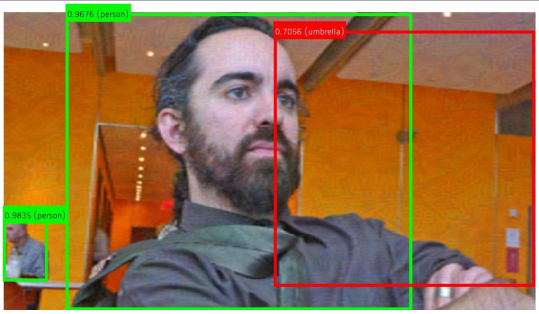
scores of the objects reported, the *TEDM* scores higher than Faster RCNN. Considerably, for objects such as the `train`, `stop sign` and `pizza`, we can clearly see a huge difference, as *TEDM* benefits from the use of visual and contextual features in contrast to Faster RCNN, which depends only on the visual features. However, in the case of `toothbrush`, where UAP perturbation is applied, Faster RCNN outperforms the *TEDM*. The reader is referred to Appendix F.2, for the AUC scores for all objects.

In addition, Table 6.5 illustrates the mAP and F1 scores for *TEDM* and Faster RCNN, where FFF and UAP perturbations are applied. Clearly, *TEDM* scores a higher performance than Faster RCNN in both cases. However, in the case of UAP attack, we can see that both models seem to be less impacted compared with FFF, which is in line with results already reported in Chapter 5.

Three images are used to present the impact of the attacks (*i.e.*, FFF and UAP) on Faster RCNN and *TEDM*, as shown in Tables 6.6 and 6.7. First, FFF perturbation is added to the entire image, as in Table 6.6. In the first image, Faster RCNN detects two objects correctly, which are two `persons`, whereas incorrectly detects an `umbrella`, which is actually a `wall`. The same perturbed image is passed into the *TEDM*, which predicts only the presence of the correct objects, which are the two `persons`. It relabels the `umbrella` as a background. However, in compare with Faster RCNN, *TEDM* scores the correctly detected objects with lower confidences, but still higher than *0.81*.

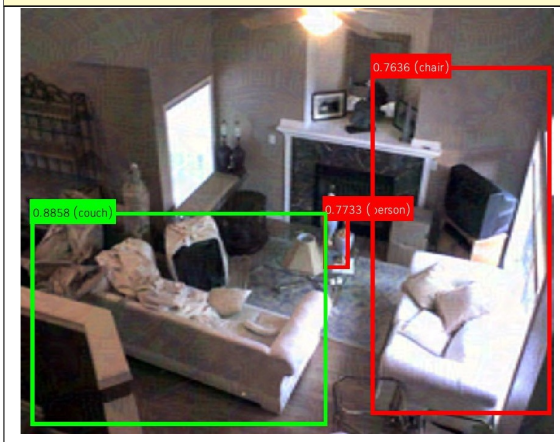
Similarly, the *TEDM* performs much better than Faster RCNN as presented in the second image. It only detects one object, which is a `couch`, with a confidence of *0.9605*. However, when Faster RCNN is applied, it detects three objects, where two are incorrectly-detected due to the perturbations added. A `person` and a `chair` are the false detections. In the last image, again, *TEDM* performs better. Faster RCNN labels a `horse` as a `person`, where *TEDM* detects the `horse` correctly with a very high confidence. However, both models incorrectly detect the `person` in the centre of the image, but *TEDM* reduces the confidence to *0.6832*.

Table 6.6.: Results: Faster RCNN and TEDM outputs for FFF perturbed images: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively.

FFF Perturbation	
Original Image	
	
Faster RCNN	TEDM
	
Original Image	
	

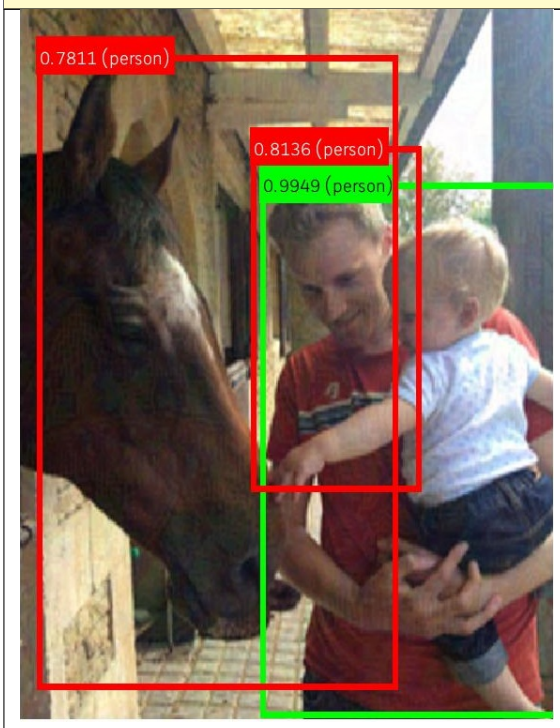
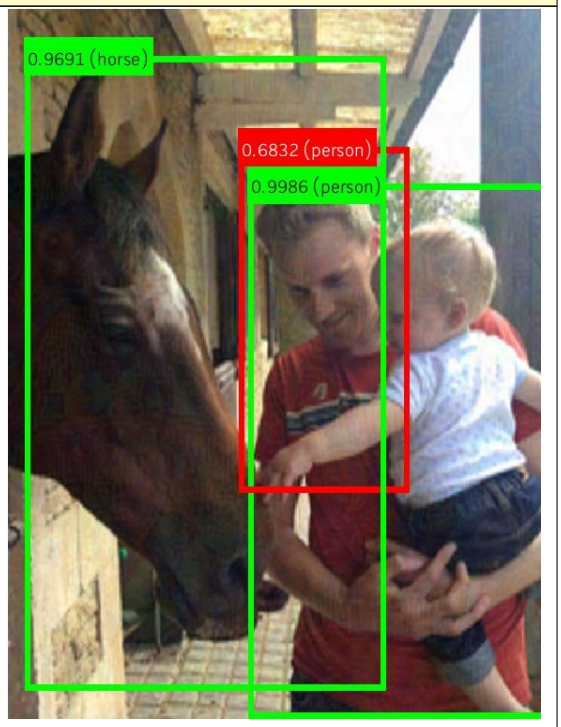
Continued on next page

Table 6.6 – continued from previous page

Faster RCNN	TEDM
	
Original Image	
	

Continued on next page

Table 6.6 – continued from previous page

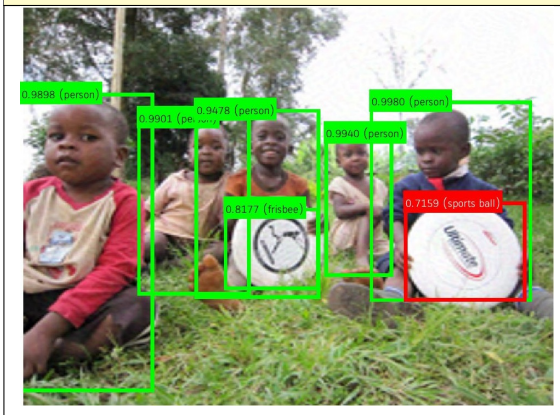
Faster RCNN	TEDM
	

In terms of UAP perturbations, three images are inputted as well, as illustrated in Table 6.7. In the first image, we can see that both models perform very well. However, Faster RCNN fails to detect one of the `frisbees`, which labels as a `sports ball`. *TEDM* correctly labels this `frisbee` with a confidence of *0.9089*. Both models score all other objects with considerably high confidences, even though perturbations are added. In the second image, a `train` is presented. Faster RCNN correctly detects the `train`, but also detects two other objects (*i.e.*, two `persons`), which are false detections. *TEDM* does only predict the `train`, and no false detection is attempted. Even one object is detected, *TEDM* predicts it with a higher probability than Faster RCNN.

Finally, in the last image, *TEDM* fails to classify one of the correctly detected objects, but also it still performs well in not detecting false detections compared to Faster RCNN. In detail, Faster RCNN detects four objects: two correct, which are a `dining table` and a `wine glass`, and two incorrect, which are two `carrots`. However, *TEDM* only detects the `wine glass`, and labels all false detections as background.

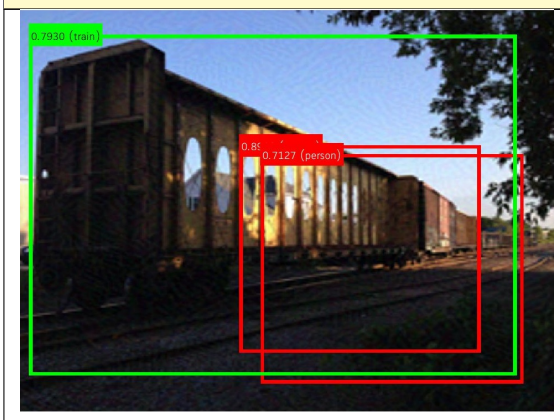
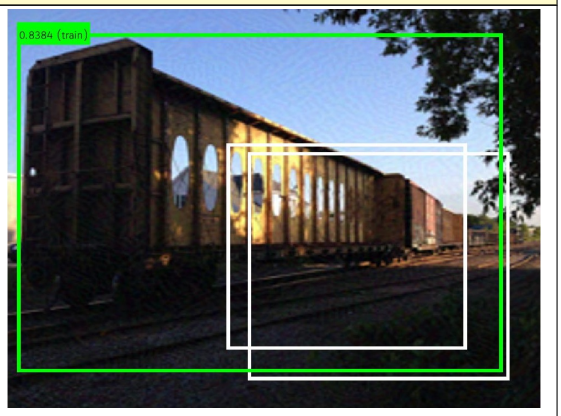
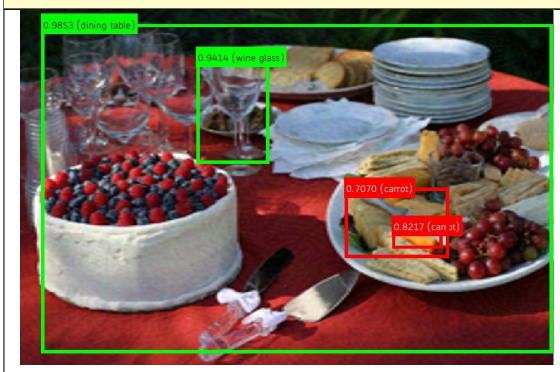
In short, we can clearly see that *TEDM* performs better scoring higher AUC scoring in the majority of cases, and higher mAP and F1 scores compared with Faster RCNN, and the *Relabelling Model* (See Section 5.3.1 in Chapter 5, for more information about the *Relabelling Model*). From the reported results, I am confident that *TEDM* is an excellent tool to overcome some of the errors that Faster RCNN attempts. Below, both models are examined when perturbations are added to regions rather than, as experimented in Chapter 5 for compatibility purpose.

Table 6.7.: Results: Faster RCNN and TEDM outputs for UAP perturbed images: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively.

UAP Perturbation	
Original Image	
	
Faster RCNN	TEDM
	
Original Image	
	

Continued on next page

Table 6.7 – continued from previous page

Faster RCNN	TEDM
	
Original Image	
	
Faster RCNN	TEDM
	

Perturbation On Regions

In this section, perturbation is added to detected regions. This is, as explored earlier in Chapter 5, impacting the performance more than it does when the perturbation is added to the entire image.

Looking at Table 6.8, we can see that *TEDM* outperforms Faster RCNN in both types of perturbation. FFF seems to impact both models lower than what UAP does, but in general, both attacks significantly drop the performances of both models. I believe *TEDM* can be offered, to some extent, yet as a solution to address perturbation issue in the field

Table 6.8.: mAP and F1 scores in percentages [%] for Faster RCNN and TEDM on perturbed regions.

	FFF		UAP	
Models	$mAP_{0.5}$	F1 Score	$mAP_{0.5}$	F1 Score
Faster RCNN	26.25%	20.78%	23.18%	18.17%
TEDM	**35.28%**	**26.25%**	**31.39%**	**24.14%**

Table 6.9.: A comparison between some MSCOCO object classes AUC scores between Faster RCNN and TEDM on perturbed regions.

	FFF		UAP	
Class Label	Faster RCNN	TEDM	Faster RCNN	TEDM
Person	0.18573	**0.31982**	0.14653	**0.28385**
Motorcycle	0.11676	**0.17328**	0.12056	**0.20871**
Train	0.05501	**0.08737**	**0.03883**	0.02912
Cat	0.06807	**0.10563**	0.03990	**0.84507**
Horse	0.08057	**0.23417**	0.13774	**0.15822**
Giraffe	0.10683	**0.30769**	0.12179	**0.26927**
Sports Ball	0.46581	**0.60370**	0.33207	**0.59401**
Bottle	0.20898	**0.28608**	0.15695	**0.27091**
Wine Glass	**0.27083**	0.19791	0.24791	**0.25625**
Donut	0.14443	**0.30263**	0.04464	**0.27310**
Bed	0.01515	**0.11940**	0	0
TV	0.09269	**0.28205**	0.09116	**0.20879**
Microwave	**0.17241**	0	0.06896	**0.09143**
Vase	0.24561	**0.28472**	0.20116	**0.24149**
Mean	0.14310	**0.23058**	0.12043	**0.19991**

of object detection. Yes, it is dependent on Faster RCNN detection outputs, but it can be end-to-end trained, and that is expected to increase the performance. This is one of the limitations of the model, which are to be discussed in Chapter 7.

AUC scores are computed as presented in Table 6.9. On average, *TEDM* performs better than Faster RCNN in the two cases of perturbation. As said, both models perform better when FFF attack is applied, unlike when UAP perturbation added. However, in some classes such as `microwave`, Faster RCNN has higher AUC scores (as per class) comparing with the proposed method. In comparison with the *Relabelling Model*, *TEDM* again achieves higher scores in terms of AUC average score, mAP and F1 scores. The reader is referred to Appendix F.3, for the AUC scores for all objects.

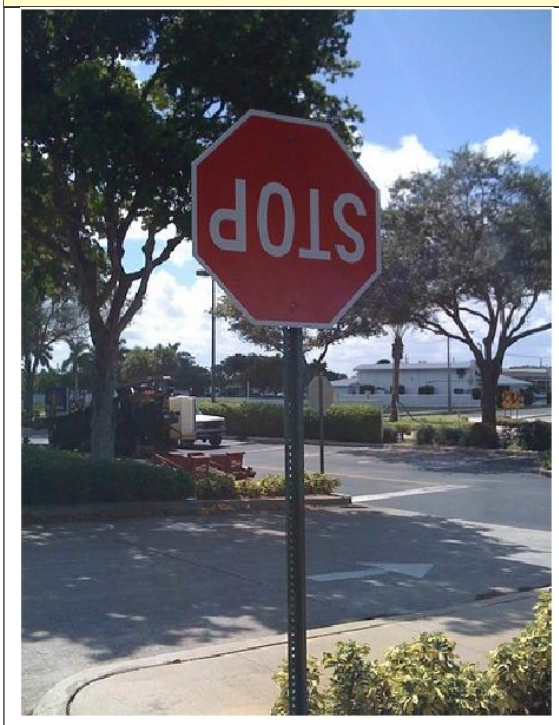
Tables 6.10 and 6.11 present some results obtained from the application of both Faster RCNN and *TEDM*. The original inputted image and the number of detected regions before any perturbation added are shown, followed by the predictions of both models when perturbations added per region. As the aim, here, is not just to illustrate the differences between models performances before and after the attack, but rather to show the differences among both models after attacking regions. Therefore, results after attacks are the only ones reported to provide ease when comparing.

Table 6.10 presents the results, where FFF perturbation is added to regions. In the first

image, we can see Faster RCNN detects three regions before the attack. However, only two are detected after, which are a `stop sign` and a `car`. The *TEDM* only detects the `stop sign`, but not the `car`, which is actually a false detection. From this result, we can notice that the *TEDM* performs better than Faster RCNN in reducing the error rate. Similarly, in the following result, the *TEDM* predicts no objects. All objects detected by Faster RCNN after attacked are false detection. Again, the *TEDM* seems to help to prevent false detections that Faster RCNN outputs. Noticeably, as shown in the detected regions before the attack, the `bed` is detected, but after the attack, it could not be predicted. This is a good example of how negatively the perturbation impacts the model, which is in line with the finding of Chapter 5 that when the regions are large in size, they are likely to be impacted by perturbation resulting to not being detected.

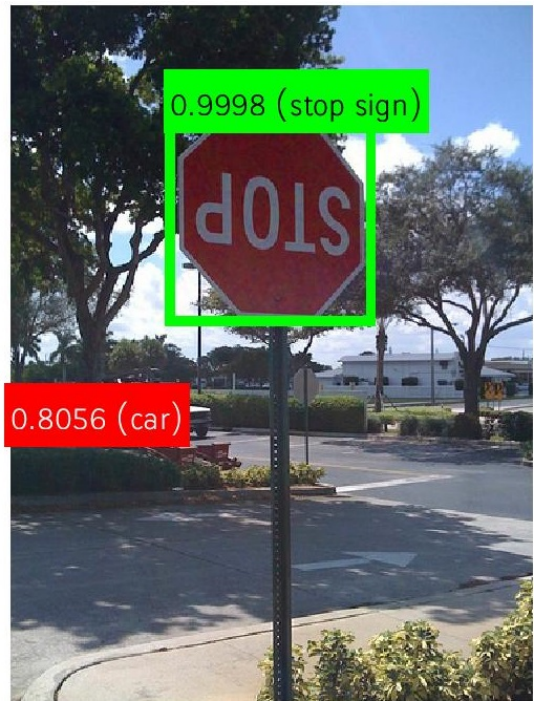
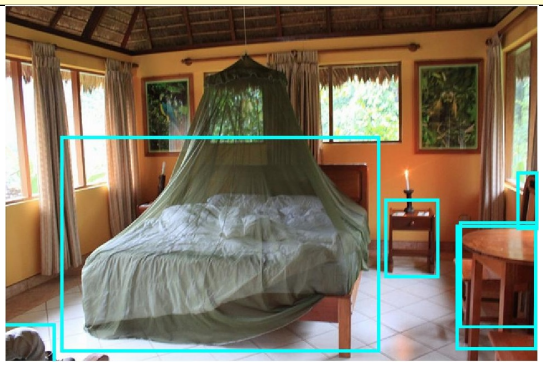
In the third image, a few numbers of `people` and two `horses` are presented. Faster RCNN detected most of the `people` and the two `horses` before perturbation is applied. However, after perturbation added to regions, only one `horse` and two `persons` are detected. As we can see, there are more obvious `people`, at least to humans, that the detector was expected to detect, such as the two `persons` riding the `horses`, which are not detected. In terms of the *TEDM* performance, It fails to detect one of the correct detections that Faster RCNN outputs, which is the `person` on the left side of the image, but still detects all others. It scores the `horse` with higher confidence than Faster RCNN, whereas it reduces the confidence of the second `person` to *0.6522*.

Table 6.10.: Results: Faster RCNN and TEDM outputs for FFF perturbed regions: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively.

FFF perturbation	
Original Image	Detected Regions Before Attack
	

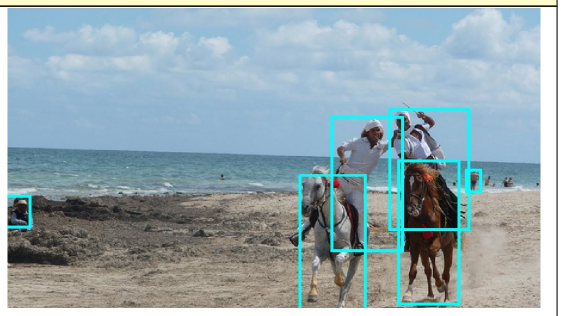
Continued on next page

Table 6.10 – continued from previous page

Faster RCNN	TEDM
	
Original Image	Detected Regions Before Attack
	
Faster RCNN	TEDM
	

Continued on next page

Table 6.10 – continued from previous page

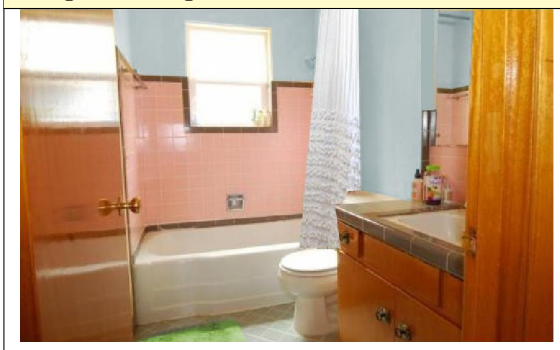
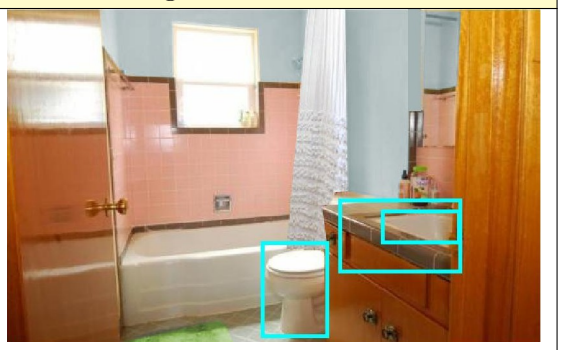
Original Image	Detected Regions Before Attack
	
Faster RCNN	**TEDM**
	

In terms of UAP perturbation, as shown Table 6.11, in the first result, Faster RCNN detects two objects correctly, which are a `toilet` and a `sink` with high confidences even regions are perturbed. *TEDM*, moreover, detects them correctly as well, and similarly with high confidences. This can show how *TEDM* can be a compatible tool to use for object detection. As shown in the regions detected before the perturbation, the `sink` is detected twice, but on a larger scale that is not predicted by Faster RCNN after the attack.

In the following image, five regions detected before perturbation, but only four are detected after. Faster RCNN detects two `cups` and a `cake` correctly, but detects a `bowl` incorrectly. Similarly, *TEDM* detects the `bowl` incorrectly but decreases its confidence significantly compared with Faster RCNN, whereas it increases all the other correct objects. In the last reported images, we can see that ten regions detected before the attack, and only half of them are detected after. Faster RCNN detects four objects correctly, but fails in detects the `cup`. The *TEDM* fails to detect the `person` that Faster RCNN already detects, but that can be due to the perturbation and the lighting conditions.

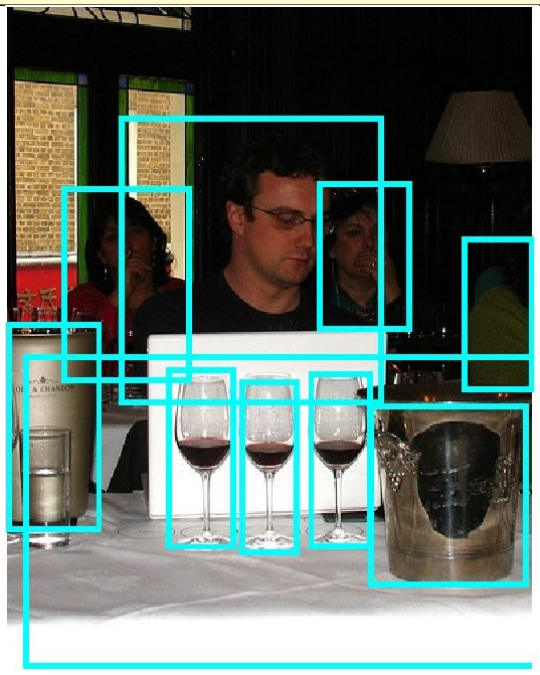
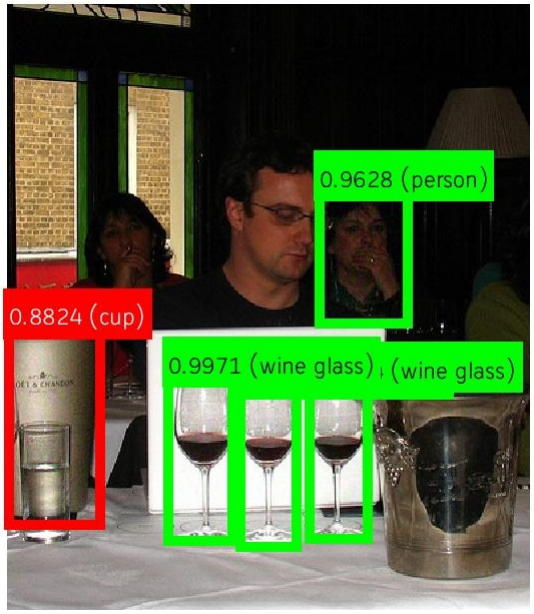
Overall, *TEDM* is performing better than Faster RCNN to tackle adversarial perturbations. This is because of the features encoding during the encoder process, as it learns the spatial and visual features.

Table 6.11.: Results: Faster RCNN and TEDM outputs for UAP perturbed regions: Green, red and white boxes represent correct detection, incorrect detection, and objects removed and relabelled as background, respectively.

UAP perturbation	
Original Image	Detected Regions Before Attack
	
Faster RCNN	TEDM
	
Original Image	Detected Regions Before Attack
	
Faster RCNN	TEDM
	

Continued on next page

Table 6.11 – continued from previous page

Original Image	Detected Regions Before Attack
	
Faster RCNN	**TEDM**
	

6.5. Summary

As stated, the impact of some perturbation models is tested on MSCOCO 2017 to examine the performance of Faster RCNN and how robust it can be when perturbation added to input images. It is reported that the detector performance dropped noticeably, where *TEDM* helps to improve the performance and reduce the false detection alarms. Two methods used to test the impact of perturbation, which are testing the entire image when the perturbation is added, whereas the second method is only adding perturbations to the detected regions and then inputted into the detector again. As experimented, the impact of perturbation seems higher when applied on regions, which can be due to the size of the regions, the larger the region is, the more it is impacted. Surprisingly, UAP perturbation affects the performance of the models when added to the entire image less than what FFF does. However, when added to regions, it drops performances considerably, which can be due to the data-dependency. In both cases *TEDM* outperforms Faster RCNN, as it helped improving the detection performance, and corrected false detections due to the inclusion of both visual and contextual features captured from images. It was also shown that *TEDM* achieves a better performance than the *Relabelling Model*, in both natural and perturbed images. However, *IRM* outperforms the performance of *TEDM*, and this can be due to the process of iterations applied to this model.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1. Summary of The Thesis

Real-world objects usually appear in particular environments among other objects. such environments are known as the context of those objects. Therefore, it is expected that knowing the context helps determining objects, even when their appearances are not fully observed.

After learning and exploring the contextual knowledge and constraints among objects in MSCOCO 2017 dataset (*i.e.*, the prime dataset used in this work), sixteen contextual relationships were proposed, as presented in Chapter 3. The proposed relationships are computed to the maximum possible variations among objects, thus the classifier applied can be trained with ease. They measure the co-occurrence, spatial and scale contexts among objects. Spatial contexts consists of four sub-groups, which are boundary, central, distance and overlapping relationships, whereas scale consists of three relationships, which are larger, smaller and equal.

Upon the proposed sixteen relationships, and the speed and accuracy of Faster RCNN detector, which was reviewed and compared with other detection methods as in Chapter 2, two novel contextual models were proposed in Chapter 4. The proposed contextual models, namely the *Rescoring Model* and the *Relabelling Model*, perform better than Faster RCNN and YOLO. Such models proved the importance of contextual information when explicitly extracted from digital scenes, where they can be applied even for out-of-context object detection. The *Relabelling Model*, as it re-rates and relabels detected objects upon the contextual knowledge, was also examined on perturbed images. As such images are likely to lack some visual features about objects, and thus the contextual information can be an important cue. Two types of perturbations were applied, namely Universal Adversarial Perturbation and Fast Feature Fool, as presented in Chapter 5. Figure 7.1 illustrates the performance of the *Relabelling Model* when perturbed images were applied.

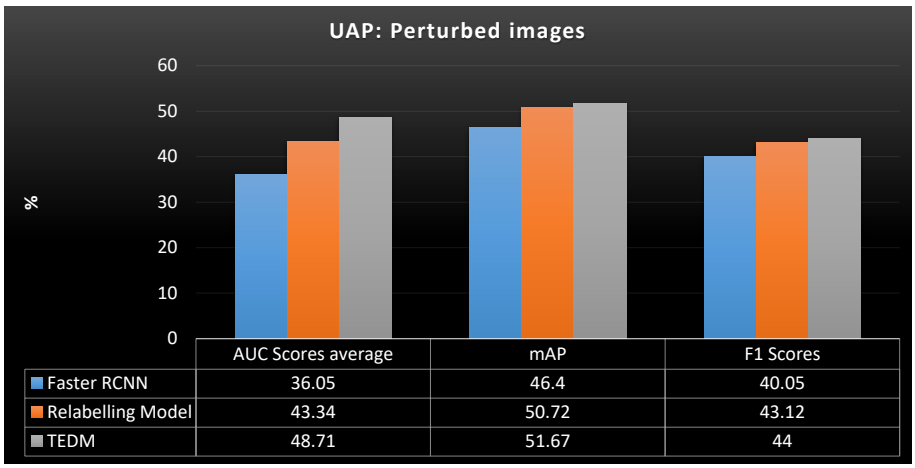


Figure 7.1.: UAP Perturbed Images: Faster RCNN, Relabelling Model Vs. TEDM

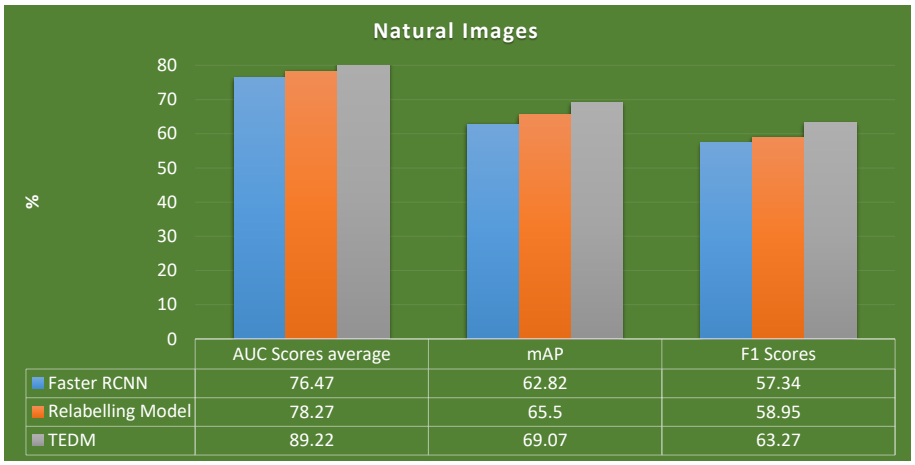


Figure 7.2.: Natural Images: Faster RCNN, Relabelling Model Vs. TEDM

In Chapter 6, a new novel method, namely Transformer-Encoder Detector Module, is proposed. This model implicitly leverages contextual information encoded from digital scenes, unlike the *Relabelling Model*. *TEDM* is built upon the use of the transformer model proposed by Vaswani et al. (2017). The performance of this model is illustrated in Figures 7.1 and 7.2 as evaluated in perturbed images and natural images, respectively.

Throughout those mentioned chapters, the objectives of this thesis were fully met and achieved. The questions set to be answered, as the final objective, have been researched and fully answered as follows. Observing from the reported statistical and visualised results, it can be surely seen that contextual information does help improving the detectors performance, where an increase in AUC scores, mAP and F1 scores was illustrated. The proposed models, which extracted contextual information implicitly and explicitly, are believed to not only rescore objects confidences, but also to relabel them, leading to an increase in the performance, and a decrease in the false detection ratio. Finally, as the last question was set, which investigates the impact of contextual information when perturbations when added. As detailed in Chapters 5 and 6, the contextual models do improve detection performance when perturbations added, because they do not only depend on appearance features, which can be directly impacted by perturbations, but also they depend on the contextual features among detected objects.

7.2. Future Work and Possible Extensions

As reported above, the proposed models have shown greater performances compared with the baseline detectors chosen in this work (*i.e.*, Faster RCNN and YOLO), as evaluated on different datasets (*e.g.*, MSCOCO 2017, PASCAL 2012). However, as we aim to teach machines to see, understand and interpret digital scenes, as we humans do is still a challenge. Therefore, challenges in computer vision are still and would remain in need for further studies and explorations until computer vision achieves human perception or even better. Such challenges are the speed of how machines can be trained, how they can see, how they can observe objects when occlusion, lighting conditions ... etc are presented, and how they can be reliable. All are needed to be explored.

Yet, contextual information, either explicitly or implicitly extracted, can be suggested as a solution. As reported, the more we understand the scene, the deeper we are likely to interpret it accurately. As we observed in Chapter 4 that information extracted from the scenes helped to build the *Rescoring Model* to re-rate detector predictions upon their context. However, when I researched further, I proposed the *Relabelling Model*, which does not only re-rate objects confidences, but also helps to relabel detector predictions when a better fit, in terms of context, is suggested. Such a model can be applied as a post-processing to most detectors. However, as it only focuses on both re-rating and relabelling, refining boundary boxes was left as a future work. Due to the time limitation and resources availability, boundary boxes refinement is left for further research and can be a direction to follow. Furthermore, rather than only depending on classification-based methods for refining predictions, apply graphical models can be also investigated. Such models can encoder relationships (*i.e.*, scene-object level, object-object level) and object attributes, thus they may improve performance and infer object states, where a simple way to visualise the structure of relationships can also be provided.

Later when perturbed images were used to test the proposed models, *TEDM* was developed, as I believed it can show higher performance, which it did. Yes, this model is built upon the success of the Transformer Model, but was only used a post-processing step for refining Faster RCNN predictions. If enough time was allowed, developing an end-to-end attention-based model would be attempted. This is expected to perform better, because this model does detect and classify objects relying on both visual and contextual features within the same pipeline. Attention mechanism could help to focus on relationships among objects, and emphasise the correlations between them, to implicitly capture semantic context. This is due to the application of the dot product of the query vector with the key vector of the respective object focused on.

Contextual information can be better learnt from multiple objects. In other words, the more objects presented in an image, the more it is likely to learn object-object relationships. This, therefore, requires datasets to include more object classes and increase the number of objects per image, thus sufficient contextual consistency can be obtained. This thesis shows the extraction of the contextual information in a supervised manner, but developing a self-supervised or zero-shot learning model can be a good direction to be looked at. This

can also reduce the load of demanding more labelled data as placing bounding boxes for every object in each sample is seen time-consuming and costs a lot. This can also help to focus on how to leverage Web images.

Appendix A

MATRICES OF CONTEXTUAL INFORMATION

This section presents matrices for each type of contextual information I propose, where statistics presented show the number of instances when relationships occur. First, as stated earlier in Section 3.5, the first category of relationships proposed is semantic, which is the meant to statistically present the likelihood of objects presenting in the same scenes together. A matrix shown the semantic relationship (co-occurrence) between all 80 objects as in MSCOCO 2017 dataset is presented in Figure A.1. The x-axis represents the objects class as names, where the y-axis represents them as numbers (*i.e.*, 1-80).

Spatial relationships, which are the second category of relationships proposed, consists of 4 sub-groups, which are 1) boundary, 2) central, 3) distance and 4) overlapping. Matrices shown the statistical analysis explored from the training dataset in MSCOCO 2017 for those relationships are presented in Figures A.2 to A.13. The first four Figures (A.2 to A.5) show the matrices for all boundary relations (*i.e.*, above, below, left, and right). The following four figures (A.6 to A.9) show the matrices for all central relations (*i.e.*, above, below, left, and right). The following two Figures (A.10 and A.11) present the distance relationships, which are near and far respectively. Similarly for the overlapping (*i.e.*, Yes and No), which are illustrated in Figures A.12 and A.13.

Finally, matrices presenting larger, equal and smaller relations, which are the sub-groups of the third proposed contextual information category (*i.e.*, scale relationship) are presented in Figures A.14 to A.16, respectively.

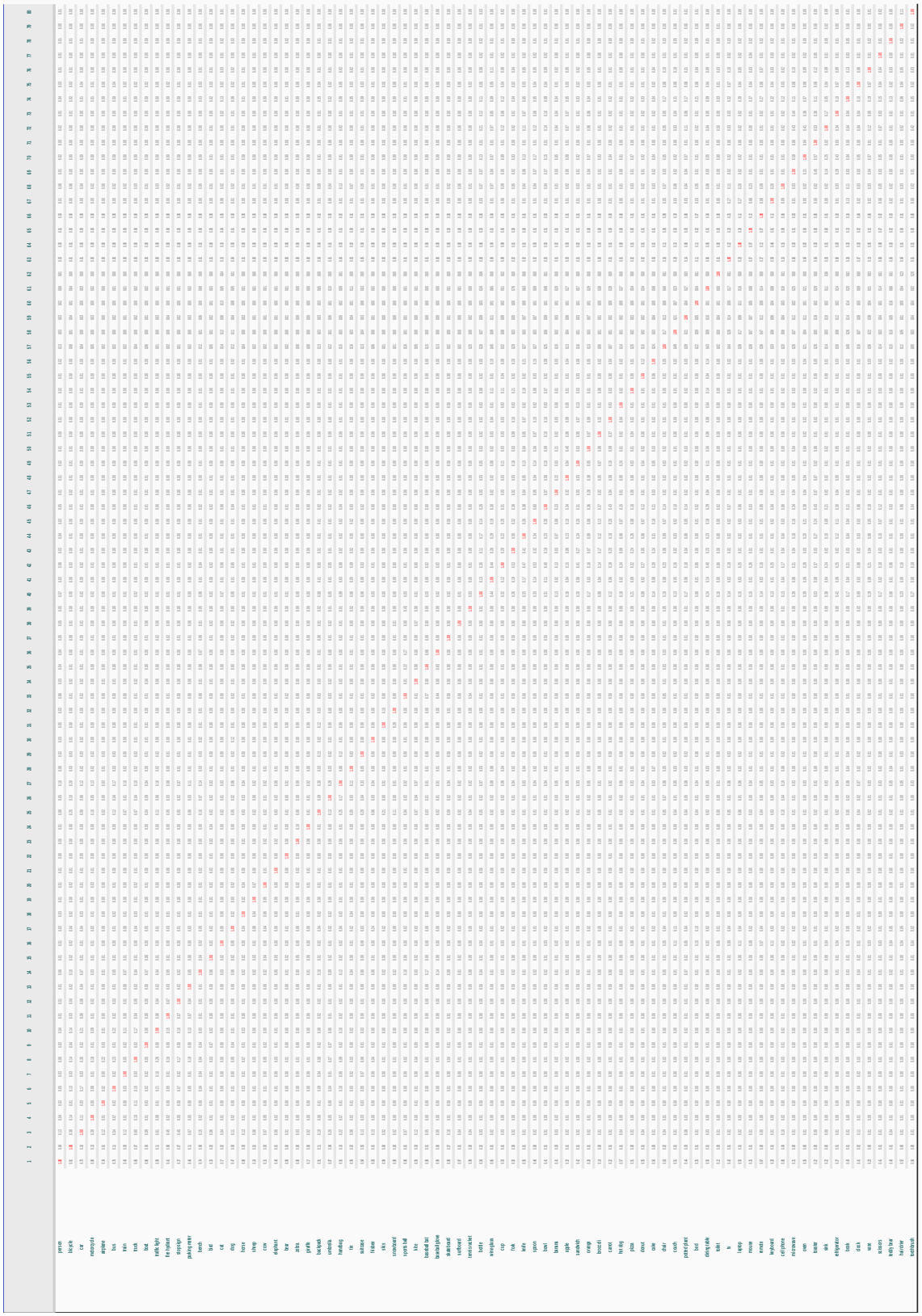


Figure A.1.: Matrix: Co-occurrence

Figure A.2.: Matrix: Boundary Above

Figure A.3.: Matrix: Boundary Below

Figure A.4.: Matrix: Boundary Left

Figure A.5.: Matrix: Boundary Right

Figure A.6.: Matrix: Central Above

Figure A.7.: Matrix: Central Below

Figure A.8.: Matrix: Central Left

Figure A.9.: Matrix: Central Right

Figure A.10.: Matrix: Near

Figure A.11.: Matrix: Far

Figure A.12.: Matrix: Overlapping

Figure A.13.: Matrix: No Overlapping

Figure A.14.: Matrix: Larger

Figure A.15.: Matrix: Equal

Figure A.16.: Matrix: Smaller

Appendix B

SCORES FROM EXAMINING RELATIONSHIPS

B.1. Further Experiments: More Classifiers Applied

B.1.1. KNN Classifier

k-nearest neighbours (KNN) classifier is examined on the *Rescoring Model* where the baseline detector is Faster RCNN. KNN is a classification method that has been widely used in many application of machine learning and data mining, which could be due to the simplicity of this algorithm and its implementation (Cover and Hart, 1967). The same steps followed in Experiment 4.2.4, in Chapter 4 are repeated here, only the classification method is changed to KNN (all the 16 relationships are encoded).

Different K values were tested for both MSCOCO 2017 and PASCAL2012, which are [k=50, 100, 200, 500]. Even though, K value is set differently, the baseline detector (*i.e.*, Faster RCNN) performs significantly better than KNN classifiers in the case of MSCOCO 2017 and PASCAL2012 datasets, as presented below.

MSCOCO 2017:

KNN classifiers, even the K value is set differently, preforms poorly compared to Faster RCNN detector as the AUC scores presented in Table B.1.

Table B.1.: AUC Scores: KNN Rescoring Model Vs Faster RCNN Detector

Model	AUC Score
KNN (k=50)	0.65937
KNN (k=100)	0.65516
KNN (k=200)	0.64618
KNN (k=500)	0.62810
Baseline Detector	**0.76472**

PASCAL2012:

Similarly, in PASCAL dataset, KNN classifiers performances are lower than the detector, as presented in Table B.2.

Table B.2.: AUC Scores: KNN Rescoring Model Vs Baseline Detector

Model	AUC Score
KNN (k=50)	0.70197
KNN (k=100)	0.69781
KNN (k=200)	0.68807
KNN (k=500)	0.60055
Baseline Detector	**0.78432**

B.1.2. BoDT Classifier

Bag of Decision Tree (BoDT) is another classifier applied on the *Rescoring Model*. First, binary decision tree classifier is applied, but due to its low performance, a BoDT is applied. BoDT can work well with large datasets, as the case of this work, but it is slow, and hence it is replaced by NN classifier, even though, it preforms slightly higher than all other classifiers including NN classifiers (Breiman et al., 1983).

First, one decision tree is applied, and then forest (*i.e.*, bag of Decision Trees) is applied, where results are shown below as per dataset.

MSCOCO 2017

Decision Tree classifier, even it is faster than BoDT, it shows very low performance, as the results shown in Table B.3. AUC scores for both Decision Tree and BoDT classifiers compared to Faster RCNN are reported. In regards to the forest classifiers, different number of trees are applied, as reported.

Table B.3.: AUC Scores: BoDT Rescoring Model Vs Baseline Detector: MSCOCO

Model	AUC Score
Decision Tree	0.65727
BoDT (50Trees)	**0.76525**
BoDT (100Trees)	**0.76919**
BoDT (200Trees)	**0.77042**
BoDT (500Trees)	**0.77180**
Baseline Detector	0.76472

PASCAL2012

Similarly, in PASCAL dataset, BoDT classifiers performances are higher than the detector as well as the decision tree classifier, as presented in Table B.4. In can be seen that the

average of AUC scores with 500trees is the highest among all other models includes the baseline detector.

Table B.4.: AUC Scores: BoDT Rescoring Model Vs Baseline Detector

Model	AUC Score
Decision Tree	0.70203
BoDT (50Trees)	**0.79005**
BoDT (100Trees)	**0.78687**
BoDT (200Trees)	**0.79163**
BoDT (500Trees)	**0.79529**
Baseline Detector	0.78432

B.2. Scores of Each Relation for Each Object Class in MSCOCO 2017

This section presents the AUC scores for all 80 classes in MSCOCO 2017, showing a comparison between one-relation model and Faster RCNN for each object. The reader is referred to Section 4.2.4, Chapter 4, for more information.

B.3. AUC Scores of Each Relation Obtained During The Five Trails

This section provides more statistical analysis, showing the AUC scores for each object class for each relationship during the five trails, to report the STD. Refer to Section 4.2.4 for more information about the use of this statistical analysis.

B.3.1. Co-occurrence Relationship

Refer to Figure B.2 for the AUC scores for Co-occurrence Relationship reported during Five Trials.

B.3.2. Boundary Relationships

Refer to Figure B.3 for the AUC scores for Boundary Relationship reported during Five Trials.

B.3.3. Central Relationships

Refer to Figure B.4 for the AUC scores for Central Relationship reported during Five Trials.

ClassName	Class_ID	Faster RCNN	Co-occurrence	Boundary	Central	Overlapping	Near/Far	Scale
person	1	0.84345527	0.838418	0.843703	0.839101	0.841264	0.838385	0.837653
bicycle	2	0.76073519	0.777326	0.741054	0.776025	0.79164	0.754392	0.777001
car	3	0.81246485	0.810231	0.808546	0.805751	0.810759	0.798097	0.802519
motorcycle	4	0.79441233	0.776628	0.795977	0.755364	0.771264	0.777586	0.785441
airplane	5	0.88300629	0.859501	0.882684	0.875307	0.877415	0.858799	0.854935
bus	6	0.89180145	0.881762	0.88483	0.8628	0.890686	0.894869	0.890407
train	7	0.93078609	0.910194	0.895328	0.890473	0.932039	0.929612	0.922027
truck	8	0.70042735	0.754939	0.71963	0.729858	0.739947	0.754659	0.714026
boat	9	0.69942281	0.693955	0.672317	0.657387	0.659548	0.682224	0.689605
traffic light	10	0.78979468	0.76788	0.797697	0.767986	0.788253	0.780083	0.797379
fire hydrant	11	0.94070081	0.886792	0.92217	0.816038	0.943396	0.875	0.931604
stop sign	12	0.93012704	0.898671	0.834718	0.912791	0.923588	0.902824	0.894518
parking mete	13	0.96590909	0.704545	0.943182	0.875	0.738636	0.931818	0.795455
bench	14	0.77859043	0.794271	0.770443	0.796354	0.801172	0.784375	0.791406
bird	15	0.80131694	0.766488	0.792063	0.781039	0.774362	0.770709	0.790362
cat	16	0.87767345	0.897951	0.880154	0.871063	0.88694	0.891549	0.906018
dog	17	0.80846395	0.829781	0.807837	0.805956	0.822257	0.813166	0.856426
horse	18	0.92223273	0.905405	0.888471	0.880431	0.893431	0.902326	0.885734
sheep	19	0.78563074	0.823015	0.830394	0.825737	0.829661	0.830394	0.822021
cow	20	0.82301666	0.829665	0.835744	0.852411	0.874109	0.857547	0.836583
elephant	21	0.74058824	0.721765	0.725412	0.743647	0.73	0.740706	0.758706
bear	22	0.94791667	0.940972	0.913194	0.840278	0.909722	0.9375	0.881944
zebra	23	0.89824561	0.873414	0.898516	0.881916	0.840756	0.877733	0.868421
giratte	24	0.85769231	0.875641	0.850641	0.821154	0.865705	0.841667	0.798077
backpack	25	0.61494551	0.702647	0.682408	0.619097	0.682408	0.719253	0.700571
umbrella	26	0.73906640	0.73856	0.749271	0.751426	0.725631	0.730511	0.751299
handbag	27	0.67471042	0.680502	0.664093	0.737452	0.681467	0.73166	0.700772
tie	28	0.84521739	0.842899	0.764638	0.81913	0.891014	0.817971	0.830725
suitcase	29	0.69537480	0.712387	0.672515	0.725146	0.750665	0.718235	0.775651
frisbee	30	0.90190972	0.926809	0.912829	0.902961	0.904605	0.904605	0.861842
skis	31	0.72606060	0.775455	0.745758	0.73697	0.75303	0.731818	0.790303
snowboard	32	0.74229691	0.7507	0.722689	0.72549	0.809524	0.736695	0.834734
sports ball	33	0.84244791	0.87066	0.853733	0.835503	0.845486	0.855035	0.868056
kite	34	0.73866150	0.750548	0.743832	0.737253	0.747533	0.755757	0.756168
baseball bat	35	0.82336956	0.882609	0.829565	0.86	0.878261	0.746957	0.883478
baseball glov	36	0.83116883	0.858998	0.836735	0.847866	0.864564	0.850649	0.855288
skateboard	37	0.84024390	0.88878	0.833415	0.896341	0.867073	0.869512	0.893659
surfboard	38	0.78483371	0.77259	0.785399	0.778926	0.8073	0.801515	0.80854
tennis racket	39	0.85895061	0.866358	0.859568	0.825617	0.862963	0.84321	0.87716
bottle	40	0.74816106	0.73732	0.754143	0.727474	0.735711	0.742795	0.721282
wine glass	41	0.77126736	0.723333	0.77875	0.741667	0.798333	0.700833	0.730833
cup	42	0.77354288	0.779086	0.77012	0.782845	0.78726	0.783591	0.788543
fork	43	0.66847826	0.706522	0.691576	0.740489	0.777174	0.736413	0.725543
knife	44	0.69595959	0.667677	0.651515	0.721212	0.655556	0.581818	0.678788
spoon	45	0.65490196	0.619608	0.788235	0.596078	0.737255	0.505882	0.65098
bowl	46	0.71389036	0.734697	0.722447	0.739072	0.741077	0.740858	0.746072
banana	47	0.68585526	0.643574	0.640707	0.641822	0.636566	0.649944	0.626533
apple	48	0.67691029	0.631579	0.654019	0.702162	0.734802	0.75561	0.668707
sandwich	49	0.76725589	0.72471	0.753317	0.672471	0.753731	0.745854	0.750829
orange	50	0.68605769	0.684936	0.66234	0.678365	0.643109	0.670192	0.66875
broccoli	51	0.63945139	0.663717	0.631389	0.629854	0.67076	0.645386	0.650894
carrot	52	0.59395424	0.611111	0.585346	0.618156	0.590781	0.600443	0.588164
hot dog	53	0.69696969	0.775401	0.558824	0.687166	0.553476	0.794118	0.558824
pizza	54	0.80429948	0.807821	0.798277	0.814122	0.819125	0.810322	0.815326
donut	55	0.73729582	0.747956	0.760757	0.730314	0.724828	0.764092	0.70213
cake	56	0.77857142	0.805844	0.798052	0.774026	0.803896	0.801948	0.785714
chair	57	0.75140872	0.763375	0.752177	0.755534	0.753077	0.749652	0.760114
couch	58	0.76153846	0.805281	0.751968	0.788271	0.839807	0.79081	0.807311
potted plant	59	0.71321424	0.750445	0.699722	0.750222	0.719967	0.769132	0.751502
bed	60	0.70948616	0.69403	0.722637	0.685323	0.738184	0.725746	0.711443
dining table	61	0.74006838	0.767631	0.733431	0.74713	0.745025	0.75132	0.756957
toilet	62	0.88879082	0.880028	0.837953	0.857001	0.858849	0.878465	0.899645
tv	63	0.86075949	0.854778	0.856934	0.855891	0.862916	0.839755	0.862568
laptop	64	0.85452562	0.864995	0.846456	0.838604	0.845365	0.811123	0.818103
mouse	65	0.86441798	0.891534	0.857143	0.923942	0.886905	0.928571	0.868386
remote	66	0.70370370	0.717706	0.693767	0.732159	0.693315	0.688799	0.699639
keyboard	67	0.75925069	0.752313	0.73173	0.775439	0.767808	0.758557	0.759251
cell phone	68	0.71807486	0.689076	0.701261	0.67605	0.742227	0.678151	0.710084
microwave	69	0.91954023	0.947619	0.961905	0.904762	0.92381	0.909524	0.947619
oven	70	0.76344827	0.769476	0.779693	0.779055	0.790549	0.750319	0.770754
sink	72	0.76167176	0.79001	0.764635	0.793207	0.791309	0.79021	0.791508
refrigerator	73	0.87063492	0.902973	0.895931	0.879499	0.892801	0.87011	0.897496
book	74	0.65153679	0.65972	0.676559	0.646151	0.64914	0.682753	0.709118
clock	75	0.92313031	0.878352	0.889937	0.895399	0.914929	0.893744	0.901357
vase	76	0.74964438	0.723718	0.742564	0.719103	0.759615	0.700769	0.727949
scissors	77	0.85714285	0.805195	0.766234	0.857143	0.896104	0.87013	0.766234
teddy bear	78	0.80049088	0.797379	0.772177	0.748656	0.779234	0.792003	0.80578
toothbrush	80	0.625	0.716667	0.658333	0.641667	0.833333	0.808333	0.7
Mean		**0.764728638**	**0.766586**	**0.758551**	**0.758882**	**0.773197**	**0.766343**	**0.766966**

Figure B.1.: AUC Scores: One-Relation Models Vs. Faster RCNN

B.3.4. Overlapping Relationships

Refer to Figure B.5 for the AUC scores for Overlapping Relationship reported during Five Trials.

	First Trial	Second Relation	Third Relation	Fourth Relation	Fifth Relation
person	0.840369	0.838087	0.839097	0.837426	0.838672
bicycle	0.770433	0.796038	0.761619	0.775641	0.786097
car	0.809099	0.811251	0.813747	0.809809	0.811146
motorcycle	0.787319	0.779870	0.755355	0.753906	0.785027
airplane	0.868003	0.864865	0.854307	0.844500	0.858974
bus	0.891892	0.894381	0.878912	0.868027	0.890411
train	0.919565	0.915653	0.907295	0.915580	0.929922
truck	0.766895	0.734683	0.746769	0.757993	0.760204
boat	0.700544	0.692050	0.687560	0.687026	0.703271
traffic light	0.783193	0.774792	0.767364	0.774177	0.768870
fire hydrant	0.901786	0.875000	0.869048	0.880319	0.901596
stop sign	0.923077	0.899798	0.944615	0.899351	0.901822
parking meter	0.700000	0.700000	0.687500	0.700000	0.737500
bench	0.826550	0.818601	0.794334	0.792528	0.794921
bird	0.765021	0.761605	0.771251	0.764569	0.765916
cat	0.899844	0.894491	0.903971	0.884987	0.886378
dog	0.834269	0.830000	0.825743	0.812308	0.808462
horse	0.908815	0.909505	0.910936	0.903894	0.903671
sheep	0.812444	0.829418	0.834425	0.826721	0.812790
cow	0.823068	0.796892	0.856414	0.826459	0.838746
elephant	0.699233	0.743186	0.707497	0.731300	0.731111
bear	0.938017	0.942149	0.940476	0.972222	0.926087
zebra	0.885315	0.878219	0.856250	0.859742	0.857807
giraffe	0.877068	0.895981	0.853147	0.866906	0.858156
backpack	0.688596	0.703283	0.674242	0.721591	0.706667
umbrella	0.755760	0.711006	0.721202	0.735212	0.746700
handbag	0.701389	0.674118	0.692857	0.684524	0.700000
tie	0.848958	0.834016	0.861538	0.853436	0.845982
suitcase	0.732258	0.722222	0.726452	0.721569	0.733987
frisbee	0.915205	0.937030	0.924951	0.914195	0.920078
skis	0.781447	0.779216	0.751020	0.769717	0.776915
snowboard	0.799342	0.759868	0.724490	0.724490	0.713333
sports ball	0.858225	0.884199	0.874459	0.867384	0.870968
kite	0.741565	0.750914	0.758119	0.741379	0.740816
baseball bat	0.894318	0.904167	0.878049	0.864895	0.885417
baseball glove	0.837209	0.885266	0.852713	0.885380	0.860819
skateboard	0.905677	0.885490	0.883159	0.884310	0.891026
surfboard	0.767510	0.752322	0.778956	0.787458	0.754935
tennis racket	0.856917	0.847826	0.879197	0.907905	0.873471
bottle	0.727581	0.729270	0.733059	0.745029	0.724039
wine glass	0.726491	0.727451	0.718402	0.729775	0.701719
cup	0.776946	0.786826	0.769652	0.781156	0.778061
fork	0.692177	0.693878	0.741497	0.765653	0.708943
knife	0.672840	0.661728	0.738916	0.664198	0.690887
spoon	0.581731	0.625000	0.620192	0.652381	0.633333
bowl	0.739064	0.733564	0.729001	0.730447	0.738348
banana	0.639851	0.657537	0.637567	0.646875	0.661508
apple	0.651081	0.615887	0.640020	0.641500	0.654000
sandwich	0.712861	0.723829	0.723330	0.762434	0.708867
orange	0.699765	0.681018	0.662950	0.687671	0.688772
broccoli	0.646012	0.669675	0.673490	0.671959	0.686995
carrot	0.602730	0.594045	0.600942	0.602488	0.607214
hot dog	0.822581	0.790909	0.754839	0.746528	0.848387
pizza	0.803620	0.821546	0.797419	0.798130	0.808371
donut	0.730815	0.722944	0.734438	0.739142	0.747459
cake	0.774359	0.792063	0.811060	0.785362	0.801075
chair	0.758666	0.767430	0.768529	0.769606	0.763941
couch	0.812963	0.779592	0.795918	0.806126	0.779278
potted plant	0.767782	0.745133	0.745249	0.752700	0.728046
bed	0.705303	0.745161	0.682576	0.744355	0.696839
dining table	0.761536	0.766242	0.758056	0.779060	0.768970
toilet	0.876812	0.880650	0.872515	0.870422	0.869527
tv	0.852237	0.851091	0.866073	0.848477	0.858041
laptop	0.860125	0.860278	0.861953	0.864936	0.865275
mouse	0.867797	0.893939	0.887560	0.907627	0.872742
remote	0.728075	0.732773	0.712493	0.702303	0.720353
keyboard	0.760402	0.753515	0.751751	0.756234	0.753968
cell phone	0.696364	0.697368	0.711538	0.689370	0.678964
microwave	0.946429	0.947090	0.941799	0.941799	0.962963
oven	0.784298	0.780769	0.764444	0.790909	0.744615
toaster	0.000000	0.000000	0.000000	0.000000	0.000000
sink	0.785644	0.784732	0.801485	0.782272	0.818059
refrigerator	0.886994	0.902116	0.903030	0.894301	0.893269
book	0.661818	0.657766	0.660760	0.662072	0.648998
clock	0.872750	0.877351	0.868737	0.867021	0.868347
vase	0.740345	0.707378	0.699083	0.736137	0.726280
scissors	0.785714	0.842857	0.800000	0.818182	0.785714
teddy bear	0.785960	0.785542	0.798977	0.786975	0.803782
hair drier	0.000000	0.000000	0.000000	0.000000	0.000000
toothbrush	0.770000	0.686869	0.690000	0.750000	0.700000
mean	0.768559	0.767253	0.764729	0.768506	0.767183

Figure B.2.: AUC Scores for Co-occurrence Relationship during Five Trials

B.3.5. Near/Far Relationships

Refer to Figure B.6 for the AUC scores for Near/Far Relationship reported during Five Trials.

B.3.6. Scale Relationships

Refer to Figure B.7 for the AUC scores for Scale Relationship reported during Five Trials.

	First Trial	Second Relation	Third Relation	Fourth Relation	Fifth Relation
person	0.844865	0.845025	0.846255	0.844099	0.845119
bicycle	0.731571	0.731170	0.736407	0.743673	0.750985
car	0.815398	0.804897	0.811512	0.806088	0.812626
motorcycle	0.782968	0.784810	0.803114	0.786172	0.788865
airplane	0.893506	0.874578	0.874890	0.870513	0.879645
bus	0.876190	0.889262	0.887415	0.896608	0.867636
train	0.891304	0.885145	0.908055	0.887918	0.883729
truck	0.711149	0.716599	0.728829	0.712597	0.701923
boat	0.662328	0.681273	0.674524	0.666598	0.675093
traffic light	0.815138	0.799452	0.797610	0.786722	0.795740
fire hydrant	0.925170	0.922872	0.917553	0.938830	0.928571
stop sign	0.825641	0.817814	0.817814	0.830972	0.847848
parking meter	0.968254	0.937500	0.937500	0.950000	0.950000
bench	0.780503	0.781130	0.761844	0.771127	0.772858
bird	0.793103	0.787300	0.804999	0.801972	0.780134
cat	0.876226	0.903438	0.876923	0.885145	0.869742
dog	0.793462	0.785385	0.811822	0.802469	0.796109
horse	0.884509	0.879039	0.868966	0.894314	0.877796
sheep	0.840000	0.818732	0.825581	0.820055	0.813725
cow	0.841577	0.847272	0.837777	0.836420	0.857086
elephant	0.761740	0.723893	0.714591	0.715232	0.730159
bear	0.895652	0.991736	0.904959	0.916667	0.900826
zebra	0.897824	0.895681	0.887427	0.901734	0.898473
giraffe	0.871429	0.857498	0.863278	0.872932	0.845324
backpack	0.658705	0.727844	0.675676	0.703565	0.689503
umbrella	0.760778	0.747197	0.762571	0.762963	0.747617
handbag	0.677156	0.693237	0.707059	0.641026	0.672454
tie	0.761612	0.743852	0.741935	0.779102	0.757514
suitcase	0.685984	0.632258	0.662745	0.678710	0.675876
frisbee	0.916179	0.926667	0.927966	0.904661	0.901623
skis	0.795015	0.723704	0.778024	0.728820	0.715926
snowboard	0.740132	0.722222	0.686667	0.690000	0.680000
sports ball	0.846877	0.851459	0.858886	0.863021	0.864191
kite	0.720837	0.742517	0.735849	0.759177	0.750165
baseball bat	0.835417	0.823920	0.838812	0.839873	0.831169
baseball glove	0.820455	0.844961	0.829457	0.825028	0.826135
skateboard	0.822823	0.817599	0.846246	0.844253	0.851863
surfboard	0.792929	0.811834	0.773131	0.785714	0.797926
tennis racket	0.855887	0.848242	0.854444	0.851481	0.848894
bottle	0.754122	0.759882	0.757501	0.751586	0.757393
wine glass	0.755486	0.772857	0.825146	0.771992	0.785294
cup	0.767349	0.773893	0.757380	0.774304	0.772361
fork	0.708408	0.720325	0.707812	0.707317	0.690625
knife	0.662562	0.637037	0.679012	0.636250	0.636700
spoon	0.771429	0.828571	0.757143	0.766667	0.769231
bowl	0.735691	0.719821	0.714946	0.716992	0.716594
banana	0.620378	0.636113	0.649414	0.641602	0.653968
apple	0.659357	0.645243	0.656109	0.628036	0.649798
sandwich	0.766393	0.759222	0.772633	0.791414	0.774177
orange	0.657277	0.661575	0.685250	0.642382	0.654403
broccoli	0.614640	0.620333	0.628501	0.639333	0.626381
carrot	0.582341	0.579851	0.596374	0.575871	0.576141
hot dog	0.548387	0.596970	0.538710	0.590323	0.533333
pizza	0.792381	0.806206	0.796606	0.799373	0.785860
donut	0.755840	0.740921	0.751316	0.774498	0.758080
cake	0.776316	0.822222	0.804274	0.772204	0.802632
chair	0.751924	0.748723	0.744757	0.752420	0.758070
couch	0.738171	0.735008	0.722135	0.771605	0.749765
potted plant	0.696241	0.691851	0.703744	0.707470	0.699712
bed	0.702273	0.688368	0.698387	0.689394	0.738710
dining table	0.723753	0.735213	0.731059	0.724971	0.741194
toilet	0.833452	0.816384	0.839649	0.834211	0.841053
tv	0.856417	0.863020	0.847389	0.864775	0.845896
laptop	0.841367	0.845833	0.854716	0.841492	0.853348
mouse	0.845497	0.850877	0.856459	0.900319	0.828070
remote	0.676487	0.689076	0.724232	0.730000	0.720395
keyboard	0.724853	0.696785	0.741821	0.725333	0.708882
cell phone	0.689744	0.703213	0.696078	0.678205	0.705624
microwave	0.957672	0.958333	0.964286	0.952381	0.957672
oven	0.794118	0.779789	0.807438	0.780998	0.757851
toaster	0.000000	0.000000	0.000000	0.000000	0.000000
sink	0.747284	0.743410	0.773810	0.764691	0.766543
refrigerator	0.923232	0.898589	0.881434	0.900000	0.882353
book	0.660484	0.663792	0.697831	0.672388	0.676012
clock	0.883754	0.873773	0.886128	0.885300	0.898058
vase	0.733926	0.717864	0.740391	0.746009	0.743192
scissors	0.772727	0.727273	0.757143	0.727273	0.727273
teddy bear	0.787952	0.771757	0.783534	0.744118	0.776707
hair drier	0.000000	0.000000	0.000000	0.000000	0.000000
toothbrush	0.718750	0.626263	0.707071	0.640000	0.690000
mean	**0.758234**	**0.756566**	**0.760209**	**0.757622**	**0.756128**

Figure B.3.: AUC Scores for Boundary Relationship during Five Trials

	First Trial	Second Relation	Third Relation	Fourth Relation	Fifth Relation
person	0.838150	0.839778	0.841093	0.840301	0.839187
bicycle	0.779940	0.765140	0.788481	0.779647	0.756611
car	0.799345	0.802164	0.804624	0.809632	0.806748
motorcycle	0.792436	0.761928	0.747497	0.750000	0.789606
airplane	0.870471	0.870940	0.884135	0.880888	0.876496
bus	0.863087	0.869048	0.879553	0.861664	0.895804
train	0.885057	0.882651	0.890819	0.883696	0.880699
truck	0.751990	0.716638	0.716123	0.748602	0.736505
boat	0.655793	0.647781	0.654477	0.662557	0.647857
traffic light	0.759760	0.759124	0.756611	0.759137	0.762525
fire hydrant	0.792553	0.813830	0.833333	0.808000	0.818452
stop sign	0.909919	0.926154	0.904858	0.902439	0.912500
parking meter	0.875000	0.875000	0.841270	0.875000	0.841270
bench	0.797774	0.793175	0.807901	0.795498	0.801997
bird	0.771513	0.786646	0.792056	0.764601	0.798246
cat	0.880376	0.867812	0.859199	0.867788	0.859194
dog	0.805029	0.827160	0.800769	0.801020	0.800760
horse	0.889559	0.876139	0.882146	0.878625	0.921651
sheep	0.832801	0.820236	0.829032	0.819343	0.834221
cow	0.846790	0.857520	0.845293	0.831928	0.855690
elephant	0.754274	0.748148	0.750036	0.720407	0.740704
bear	0.808696	0.826446	0.808696	0.837302	0.808696
zebra	0.872320	0.877843	0.875835	0.868571	0.876063
giraffe	0.802600	0.810526	0.855004	0.867854	0.810875
backpack	0.648496	0.585795	0.611742	0.628536	0.588972
umbrella	0.788067	0.748272	0.753221	0.755760	0.748534
handbag	0.747086	0.748235	0.784706	0.762731	0.757143
tie	0.822000	0.795094	0.852034	0.832650	0.822917
suitcase	0.696078	0.715686	0.726790	0.742985	0.714194
frisbee	0.901111	0.908722	0.886940	0.903509	0.904665
skis	0.760000	0.751111	0.723003	0.751116	0.723562
snowboard	0.766667	0.740132	0.725490	0.715686	0.750000
sports ball	0.864721	0.839718	0.814394	0.804864	0.821875
kite	0.740646	0.729160	0.737686	0.730528	0.740040
baseball bat	0.861573	0.859375	0.851732	0.866002	0.857900
baseball glove	0.854928	0.840936	0.862680	0.881818	0.840909
skateboard	0.891941	0.909483	0.894410	0.907509	0.880435
surfboard	0.766644	0.788136	0.769661	0.761864	0.769580
tennis racket	0.808486	0.808411	0.808889	0.819572	0.832963
bottle	0.733286	0.729436	0.729536	0.722687	0.727237
wine glass	0.740686	0.727503	0.734804	0.720425	0.745588
cup	0.780238	0.780675	0.785625	0.778473	0.778943
fork	0.700680	0.765306	0.713821	0.770408	0.728086
knife	0.709877	0.677419	0.707500	0.727833	0.716749
spoon	0.585714	0.620192	0.600962	0.604762	0.576190
bowl	0.722782	0.751630	0.752971	0.735823	0.749423
banana	0.649802	0.656869	0.631196	0.607540	0.648308
apple	0.682252	0.669556	0.675607	0.713424	0.693313
sandwich	0.666493	0.666502	0.685963	0.666002	0.688345
orange	0.695065	0.677495	0.679005	0.676643	0.682139
broccoli	0.629791	0.639438	0.620690	0.622556	0.621379
carrot	0.613591	0.601093	0.592060	0.609429	0.613343
hot dog	0.690323	0.690323	0.674194	0.719355	0.690972
pizza	0.822951	0.816195	0.807683	0.795849	0.816162
donut	0.708026	0.718947	0.718812	0.741987	0.731749
cake	0.764209	0.779570	0.784127	0.759314	0.810033
chair	0.758759	0.746458	0.760568	0.753465	0.754030
couch	0.776928	0.792894	0.772379	0.780691	0.767385
potted plant	0.750409	0.752006	0.746337	0.747779	0.734313
bed	0.704839	0.684091	0.646774	0.669355	0.697112
dining table	0.758929	0.749212	0.741943	0.748582	0.743907
toilet	0.860292	0.853896	0.855773	0.855012	0.863097
tv	0.846255	0.848414	0.851356	0.860100	0.857433
laptop	0.833422	0.833559	0.820556	0.840837	0.873183
mouse	0.933812	0.919540	0.914673	0.913793	0.946634
remote	0.722039	0.742222	0.712631	0.748889	0.724766
keyboard	0.799335	0.775510	0.782292	0.775581	0.782596
cell phone	0.694026	0.676213	0.689628	0.681649	0.661920
microwave	0.880952	0.904762	0.894180	0.904762	0.894180
oven	0.813679	0.777692	0.797037	0.739231	0.765385
toaster	0.000000	0.000000	0.000000	0.000000	0.000000
sink	0.767765	0.800590	0.794900	0.790712	0.768293
refrigerator	0.865385	0.877885	0.869697	0.880515	0.874081
book	0.647237	0.654282	0.637118	0.659382	0.645788
clock	0.898416	0.900760	0.896072	0.915590	0.897504
vase	0.729760	0.705524	0.730769	0.719645	0.723253
scissors	0.833333	0.833333	0.857143	0.833333	0.857143
teddy bear	0.737274	0.741935	0.765126	0.757028	0.730835
hair drier	0.000000	0.000000	0.000000	0.000000	0.000000
toothbrush	0.616162	0.676768	0.636364	0.696970	0.718750
mean	**0.758481**	**0.757572**	**0.756501**	**0.759063**	**0.759070**

Figure B.4.: AUC Scores for Central Relationship during Five Trials

	First Trial	Second Relation	Third Relation	Fourth Relation	Fifth Relation
person	0.841640	0.839171	0.841846	0.841738	0.840467
bicycle	0.807492	0.785461	0.789790	0.776149	0.793612
car	0.809297	0.816835	0.812293	0.813186	0.812948
motorcycle	0.754423	0.773593	0.768014	0.770229	0.775083
airplane	0.866750	0.887621	0.878446	0.876190	0.880342
bus	0.897619	0.877551	0.899716	0.886279	0.888780
train	0.945029	0.928655	0.922872	0.936957	0.938821
truck	0.731671	0.733276	0.736068	0.752530	0.732313
boat	0.658758	0.668630	0.672448	0.660158	0.654334
traffic light	0.796207	0.801541	0.774318	0.783192	0.788696
fire hydrant	0.957447	0.957447	0.937500	0.936170	0.937500
stop sign	0.912955	0.927928	0.919028	0.915992	0.950769
parking meter	0.725000	0.725000	0.737500	0.714286	0.750000
bench	0.780013	0.810754	0.813990	0.797375	0.806349
bird	0.765744	0.795994	0.785127	0.777597	0.760714
cat	0.895469	0.885303	0.897500	0.877876	0.896755
dog	0.820769	0.823045	0.848077	0.834851	0.813462
horse	0.889595	0.900373	0.895816	0.891276	0.888956
sheep	0.824795	0.827611	0.844891	0.818567	0.846880
cow	0.878215	0.866702	0.868209	0.866493	0.864802
elephant	0.735125	0.754965	0.723481	0.747712	0.724872
bear	0.979339	0.900826	0.896694	0.886957	0.908730
zebra	0.822612	0.826186	0.834718	0.841694	0.852658
giraffe	0.866824	0.853428	0.877069	0.871583	0.855398
backpack	0.693558	0.668561	0.692933	0.660952	0.653509
umbrella	0.732558	0.710112	0.727818	0.727039	0.726562
handbag	0.723776	0.663194	0.709790	0.710956	0.690476
tie	0.883929	0.901876	0.900956	0.873512	0.878066
suitcase	0.760000	0.750980	0.756774	0.769894	0.771220
frisbee	0.895712	0.892060	0.895677	0.897556	0.898635
skis	0.731111	0.743343	0.755556	0.751017	0.779134
snowboard	0.784314	0.792763	0.790850	0.792763	0.790850
sports ball	0.835938	0.833333	0.829688	0.846875	0.833874
kite	0.755941	0.762878	0.738779	0.735353	0.740816
baseball bat	0.865323	0.863787	0.879109	0.869565	0.877706
baseball glove	0.870432	0.879545	0.839372	0.878409	0.853801
skateboard	0.879880	0.859159	0.880647	0.858559	0.877019
surfboard	0.808983	0.806589	0.804233	0.793981	0.814576
tennis racket	0.849802	0.863636	0.865061	0.848242	0.867362
bottle	0.741200	0.730841	0.728803	0.738556	0.729125
wine glass	0.775340	0.812229	0.816558	0.839326	0.812957
cup	0.797250	0.779900	0.778931	0.785142	0.788696
fork	0.809524	0.771020	0.778912	0.778125	0.739796
knife	0.670370	0.635802	0.667500	0.675000	0.651478
spoon	0.757143	0.807692	0.738095	0.719048	0.735294
bowl	0.743443	0.739515	0.749718	0.730963	0.728923
banana	0.638644	0.627670	0.609445	0.642955	0.612285
apple	0.717949	0.726496	0.741458	0.710020	0.727690
sandwich	0.763210	0.756231	0.777667	0.758709	0.757071
orange	0.641373	0.654343	0.632629	0.638889	0.655112
broccoli	0.695458	0.666333	0.692549	0.654467	0.681632
carrot	0.589038	0.592151	0.580846	0.575682	0.592648
hot dog	0.562500	0.560606	0.510417	0.542424	0.566667
pizza	0.810875	0.837104	0.827376	0.830140	0.793213
donut	0.721421	0.705083	0.720887	0.721781	0.714343
cake	0.783333	0.785714	0.809211	0.811678	0.808495
chair	0.746862	0.753201	0.759059	0.756368	0.756940
couch	0.827469	0.841806	0.844136	0.831481	0.842284
potted plant	0.732405	0.710551	0.720731	0.723106	0.724534
bed	0.722873	0.766935	0.767369	0.734091	0.744731
dining table	0.728167	0.733952	0.744968	0.740840	0.729920
toilet	0.870000	0.852679	0.859287	0.878066	0.851622
tv	0.867047	0.863332	0.866627	0.859307	0.850076
laptop	0.838902	0.832892	0.834216	0.842691	0.844082
mouse	0.904503	0.918644	0.880131	0.878489	0.880508
remote	0.689076	0.683333	0.678180	0.690017	0.686852
keyboard	0.755394	0.771593	0.793780	0.759302	0.767442
cell phone	0.757740	0.752932	0.732308	0.733230	0.755385
microwave	0.915344	0.917241	0.915344	0.910714	0.924138
oven	0.780543	0.780543	0.788522	0.776973	0.806184
toaster	0.000000	0.000000	0.000000	0.000000	0.000000
sink	0.807589	0.782641	0.814486	0.796255	0.790342
refrigerator	0.889706	0.898897	0.897978	0.880769	0.883654
book	0.644207	0.657427	0.651901	0.627351	0.645445
clock	0.925294	0.920417	0.915655	0.927971	0.909216
vase	0.760407	0.762285	0.768431	0.756829	0.761346
scissors	0.893939	0.885714	0.885714	0.893939	0.909091
teddy bear	0.771757	0.767873	0.775210	0.755882	0.789409
hair drier	0.000000	0.000000	0.000000	0.000000	0.000000
toothbrush	0.808081	0.880000	0.878788	0.802083	0.870000
mean	**0.773618**	**0.773542**	**0.774706**	**0.769980**	**0.772943**

Figure B.5.: AUC Scores for Overlapping Relationship during Five Trials

	First Trial	Second Relation	Third Relation	Fourth Relation	Fifth Relation
person	0.842034	0.839878	0.837343	0.836700	0.835237
bicycle	0.761323	0.744492	0.754898	0.776042	0.771224
car	0.797586	0.798641	0.798926	0.806787	0.795659
motorcycle	0.773526	0.767547	0.759141	0.778288	0.776840
airplane	0.850307	0.847551	0.860606	0.866976	0.845726
bus	0.901500	0.896159	0.893999	0.878023	0.905442
train	0.922101	0.943641	0.929551	0.936596	0.928394
truck	0.758333	0.754252	0.736198	0.744939	0.743367
boat	0.693826	0.675723	0.684207	0.677178	0.687638
traffic light	0.784815	0.792136	0.776058	0.781478	0.766576
fire hydrant	0.861702	0.860119	0.910714	0.857143	0.869048
stop sign	0.891892	0.885417	0.896761	0.901822	0.921922
parking meter	0.925000	0.925000	0.962500	0.925000	0.937500
bench	0.799120	0.771337	0.776729	0.778696	0.784008
bird	0.763236	0.754231	0.752525	0.768359	0.767288
cat	0.889660	0.912923	0.900156	0.894592	0.892643
dog	0.793267	0.816210	0.823366	0.815563	0.793663
horse	0.901038	0.917150	0.906124	0.902310	0.906151
sheep	0.839412	0.840781	0.832674	0.836571	0.826166
cow	0.867229	0.862269	0.851852	0.860412	0.847837
elephant	0.740990	0.745244	0.731700	0.726830	0.738636
bear	0.940476	0.940476	0.929752	0.933884	0.944444
zebra	0.863636	0.865335	0.866944	0.889867	0.880355
giraffe	0.827423	0.853759	0.847482	0.842999	0.842806
backpack	0.704588	0.704588	0.709900	0.720328	0.740317
umbrella	0.739415	0.724280	0.729707	0.729296	0.726308
handbag	0.729412	0.739286	0.720000	0.723529	0.710648
tie	0.808036	0.842713	0.814165	0.823333	0.817623
suitcase	0.767241	0.686658	0.755740	0.682353	0.730719
frisbee	0.918033	0.905556	0.897246	0.922669	0.931774
skis	0.764053	0.755926	0.739259	0.723333	0.758889
snowboard	0.744898	0.779605	0.720395	0.740132	0.745098
sports ball	0.846684	0.857825	0.855729	0.842454	0.843560
kite	0.745899	0.747301	0.745951	0.757261	0.754455
baseball bat	0.748616	0.730647	0.754872	0.777409	0.747614
baseball glove	0.868182	0.847176	0.840909	0.890909	0.849391
skateboard	0.855707	0.849512	0.881135	0.874775	0.866770
surfboard	0.793957	0.816162	0.820174	0.815101	0.801356
tennis racket	0.840671	0.837538	0.835185	0.854074	0.835556
bottle	0.744671	0.743384	0.745830	0.745008	0.746431
wine glass	0.705392	0.652056	0.677958	0.725980	0.690700
cup	0.784114	0.780386	0.790476	0.780599	0.794216
fork	0.760204	0.761905	0.744715	0.762602	0.785714
knife	0.586420	0.583127	0.571960	0.564039	0.566502
spoon	0.542857	0.523810	0.576190	0.466667	0.557692
bowl	0.734330	0.740472	0.727713	0.740278	0.747500
banana	0.626475	0.653041	0.667383	0.667460	0.648638
apple	0.767500	0.753036	0.748631	0.756600	0.757085
sandwich	0.729811	0.721675	0.742976	0.727971	0.731530
orange	0.666862	0.657864	0.657462	0.648083	0.671429
broccoli	0.642602	0.664658	0.664485	0.640111	0.664356
carrot	0.607444	0.593052	0.595628	0.589552	0.586062
hot dog	0.824242	0.821212	0.812903	0.780645	0.791667
pizza	0.824302	0.812233	0.815549	0.820136	0.803148
donut	0.773979	0.768816	0.755952	0.765364	0.765103
cake	0.815079	0.795238	0.796875	0.783159	0.790323
chair	0.747602	0.753003	0.756083	0.753678	0.744023
couch	0.797222	0.768542	0.804938	0.780384	0.814757
potted plant	0.782130	0.771333	0.776155	0.764196	0.758792
bed	0.736144	0.742742	0.734709	0.743169	0.721774
dining table	0.743915	0.751481	0.750766	0.757141	0.747869
toilet	0.877119	0.880712	0.869565	0.875776	0.861260
tv	0.839969	0.823524	0.847013	0.832737	0.843264
laptop	0.815397	0.829222	0.808599	0.802383	0.809584
mouse	0.929348	0.915435	0.927432	0.932677	0.929825
remote	0.676779	0.695000	0.704310	0.691667	0.674021
keyboard	0.757075	0.765432	0.745011	0.784864	0.756652
cell phone	0.691983	0.688291	0.694924	0.703044	0.657922
microwave	0.899471	0.899471	0.899471	0.904762	0.944828
oven	0.769654	0.748889	0.789855	0.734615	0.735910
toaster	0.000000	0.000000	0.000000	0.000000	0.000000
sink	0.796760	0.788657	0.785714	0.786914	0.791832
refrigerator	0.867725	0.865385	0.862500	0.858654	0.882828
book	0.676968	0.688267	0.678678	0.671313	0.678189
clock	0.887889	0.887631	0.890979	0.890891	0.896243
vase	0.727316	0.710830	0.684194	0.702198	0.710674
scissors	0.863636	0.878788	0.857143	0.942857	0.878788
teddy bear	0.791597	0.784893	0.789960	0.772325	0.784422
hair drier	0.000000	0.000000	0.000000	0.000000	0.000000
toothbrush	0.797980	0.780000	0.830000	0.780000	0.840000
mean	**0.768435**	**0.765982**	**0.767766**	**0.766544**	**0.767877**

Figure B.6.: AUC Scores for Near/Far Relationship during Five Trials

	First Trial	Second Relation	Third Relation	Fourth Relation	Fifth Relation
person	0.837618	0.838120	0.835861	0.838654	0.835707
bicycle	0.752731	0.758471	0.782112	0.782920	0.801589
car	0.803955	0.803872	0.804533	0.811520	0.797809
motorcycle	0.758742	0.783654	0.791139	0.787244	0.784416
airplane	0.834199	0.840260	0.865419	0.849145	0.879076
bus	0.900340	0.885491	0.894381	0.890469	0.894737
train	0.917760	0.911232	0.924012	0.913978	0.923478
truck	0.700236	0.717949	0.712086	0.687247	0.736364
boat	0.688180	0.677172	0.698551	0.683471	0.681486
traffic light	0.788991	0.790922	0.799803	0.792398	0.792203
fire hydrant	0.944149	0.940476	0.944000	0.928571	0.925532
stop sign	0.878125	0.881579	0.884615	0.900513	0.889583
parking meter	0.857143	0.775000	0.714286	0.800000	0.873016
bench	0.790857	0.792146	0.782727	0.775448	0.776362
bird	0.771429	0.806474	0.789707	0.780497	0.773948
cat	0.911477	0.893055	0.922804	0.893315	0.899799
dog	0.857872	0.860990	0.841003	0.860423	0.856715
horse	0.882420	0.885714	0.890867	0.879433	0.880903
sheep	0.834248	0.827748	0.822393	0.811810	0.827845
cow	0.825489	0.841017	0.849205	0.837148	0.850223
elephant	0.739749	0.765723	0.765589	0.748063	0.754544
bear	0.892857	0.876033	0.884921	0.895652	0.904959
zebra	0.849007	0.869071	0.853964	0.871224	0.858029
giraffe	0.824271	0.838442	0.801988	0.782112	0.788346
backpack	0.712131	0.714912	0.699812	0.727904	0.719199
umbrella	0.755794	0.756859	0.755129	0.750865	0.748456
handbag	0.664336	0.700000	0.748810	0.703571	0.685714
tie	0.822511	0.829004	0.838384	0.811688	0.831169
suitcase	0.786275	0.771684	0.766578	0.753827	0.843501
frisbee	0.852941	0.887588	0.841785	0.849903	0.865112
skis	0.756215	0.797247	0.773438	0.768036	0.788354
snowboard	0.849673	0.840000	0.870000	0.828947	0.833333
sports ball	0.866359	0.877454	0.879672	0.863542	0.866871
kite	0.743707	0.748639	0.761735	0.755608	0.760121
baseball bat	0.876042	0.870626	0.866002	0.892857	0.901440
baseball glove	0.845455	0.846784	0.884211	0.864327	0.855682
skateboard	0.898581	0.881868	0.895319	0.908634	0.897398
surfboard	0.802357	0.796296	0.783559	0.808222	0.797619
tennis racket	0.907875	0.877426	0.877037	0.878756	0.884557
bottle	0.726693	0.717737	0.730204	0.723335	0.713609
wine glass	0.716991	0.742669	0.736765	0.722472	0.745588
cup	0.795311	0.777266	0.782891	0.788906	0.797374
fork	0.707812	0.730081	0.715564	0.739837	0.712585
knife	0.650246	0.676543	0.755051	0.692118	0.650617
spoon	0.700980	0.644231	0.634615	0.634615	0.609524
bowl	0.753350	0.750923	0.762746	0.746234	0.735203
banana	0.598544	0.641387	0.609127	0.637687	0.616247
apple	0.672500	0.665992	0.648567	0.662142	0.674000
sandwich	0.766162	0.768443	0.760718	0.724898	0.758225
orange	0.646322	0.708023	0.679648	0.665883	0.655321
broccoli	0.651713	0.642667	0.668108	0.654689	0.664996
carrot	0.614640	0.605211	0.588586	0.575509	0.597615
hot dog	0.579861	0.513889	0.561290	0.606452	0.600694
pizza	0.829538	0.821256	0.824434	0.808888	0.823077
donut	0.700921	0.696494	0.685897	0.696238	0.700419
cake	0.793395	0.775493	0.772204	0.804762	0.769737
chair	0.769790	0.761331	0.762457	0.753728	0.778846
couch	0.816667	0.798593	0.810675	0.791888	0.791975
potted plant	0.771331	0.761603	0.740310	0.767039	0.756454
bed	0.719697	0.688636	0.734582	0.687744	0.694770
dining table	0.754548	0.741085	0.752992	0.752199	0.755001
toilet	0.910973	0.903383	0.893861	0.912076	0.900526
tv	0.864329	0.855931	0.873466	0.871083	0.853734
laptop	0.829625	0.792740	0.828390	0.823888	0.804555
mouse	0.867816	0.858054	0.853270	0.862578	0.863636
remote	0.716009	0.690789	0.724232	0.682913	0.687778
keyboard	0.781451	0.759597	0.740029	0.753515	0.744318
cell phone	0.698658	0.710607	0.702857	0.710268	0.723426
microwave	0.968254	0.957672	0.941799	0.940476	0.958333
oven	0.799528	0.777526	0.783019	0.790370	0.751240
toaster	0.000000	0.000000	0.000000	0.000000	0.000000
sink	0.792806	0.784676	0.794074	0.777039	0.777477
refrigerator	0.892308	0.938462	0.890653	0.903493	0.875267
book	0.700545	0.696432	0.726032	0.709870	0.704424
clock	0.902361	0.895358	0.906763	0.893996	0.898261
vase	0.712676	0.727700	0.725583	0.733177	0.728013
scissors	0.772727	0.772727	0.772727	0.727273	0.772727
teddy bear	0.808434	0.807063	0.835341	0.796388	0.814860
hair drier	0.000000	0.000000	0.000000	0.000000	0.000000
toothbrush	0.656566	0.740000	0.646465	0.696970	0.666667
mean	**0.767427**	**0.766941**	**0.768168**	**0.764882**	**0.767404**

Figure B.7.: AUC Scores for Scale Relationship during Five Trials

Appendix C

RESCORING MODEL STATISTICAL RESULTS

C.1. Experiment: Applying Rescoring Model On Faster RCNN: PASCAL 2012

PASCAL 2012 dataset is used to examines the impact of the Rescoring Model, as discussed in Chapter 4, Section 4.2.4. Table C.1 shows the AUC scores for all 20 objects in PASCAL2012.

C.2. Experiment: Applying Rescoring Model On YOLOv1

As explained in Chapter 4, Section 4.2.4, YOLO is used as another baseline detector to examine the impact of the proposed *Rescoring Model*, where the AUC scores for objects detected in MSCOCO 2017 is presented in Figure C.2. Please note only the AUC scores for objects, which have been at least detected once, are shown. In other words, some objects such as *backpack*, *handbag*, *knife*, *spoon* and *toothbrush* have never been detected during testing YOLO on MSCOCO 2017 validation dataset, and thus they are not included for ease in reporting results.

ClassName	Class_ID	Faster RCNN	**Rescoring Model**
'person'	1	0.935714286	0.928571429
'bicycle'	2	0.648295455	0.734469697
'car'	3	0.816377171	0.817617866
'motorbike'	4	0.809285714	0.753571429
'aeroplane'	5	0.809949239	0.809069374
'bus'	6	0.925694444	0.854166667
'train'	7	0.783397129	0.827607656
'boat'	8	0.814041746	0.782732448
'bird'	9	0.754878981	0.824968153
'cat'	10	0.81509434	0.888050314
'dog'	11	0.686889017	0.64893617
'horse'	12	0.814430014	0.8
'sheep'	13	0.528262767	0.692055871
'cow'	14	0.687562438	0.698301698
'bottle'	15	0.799614918	0.83236177
'chair'	16	0.730226151	0.772739448
'sofa'	17	0.842156387	0.886352598
'potted_plant'	18	0.813099708	0.838130997
'dining_table'	19	0.828571429	0.704761905
'tvmonitor'	20	0.842973602	0.779454581
Mean		0.784325747	**0.793696004**

Figure C.1.: AUC scores obtained from Rescoring Model Vs. Faster RCNN on PASCAL2012

ClassName	Class_ID	Faster RCNN	**Rescoring Model**
person	1	0.829953259	0.8343488
bicycle	2	0.819088319	0.779202279
car	3	0.791131372	0.792075333
motorcycle	4	0.689678742	0.579630895
airplane	5	0.861163227	0.848030019
bus	6	0.825221239	0.827433628
train	7	0.859195402	0.83908046
truck	8	0.81686747	0.86184739
boat	9	0.803761755	0.812539185
traffic light	10	0.637506548	0.651388161
fire hydrant	11	0.758139535	0.772093023
stop sign	12	0.7375	0.7125
parking meter	13	0.875	0.75
bench	14	0.712820513	0.77008547
bird	15	0.765789474	0.748245614
cat	16	0.796825397	0.807936508
dog	17	0.76881068	0.746359223
horse	18	0.829268293	0.812195122
sheep	19	0.726858513	0.716067146
cow	20	0.869105691	0.867750678
elephant	21	0.712315462	0.713286713
bear	22	0.696428571	0.571428571
zebra	23	0.895127993	0.896779521
umbrella	26	0.799953315	0.784080299
tie	28	0.839	0.801
suitcase	29	0.85617284	0.863580247
frisbee	30	0.864864865	0.918918919
skis	31	0.801005747	0.793103448
snowboard	32	0.7	0.8
sports ball	33	0.704894083	0.677867056
kite	34	0.736081994	0.729326811
baseball bat	35	0.648351648	0.681318681
baseball glove	36	0.692307692	0.672064777
skateboard	37	0.824660103	0.80403188
surfboard	38	0.762276786	0.753348214
tennis racket	39	0.693379791	0.68466899
bottle	40	0.785057471	0.784195402
wine glass	41	0.602222222	0.653333333
cup	42	0.618079289	0.616883117
fork	43	0.361111111	0.5
bowl	46	0.685263796	0.68405094
banana	47	0.700411523	0.724279835
apple	48	0.681818182	0.545454545
sandwich	49	0.763636364	0.767272727
orange	50	0.596749811	0.58276644
broccoli	51	0.576714514	0.542902711
carrot	52	0.695652174	0.748283753
hot dog	53	0.805555556	0.824074074
pizza	54	0.726851852	0.679012346
donut	55	0.687571922	0.691887227
cake	56	0.766081871	0.78128655
chair	57	0.750159994	0.73181398
couch	58	0.786394558	0.76462585
potted plant	59	0.684489796	0.73877551
bed	60	0.805555556	0.796296296
dining table	61	0.719333333	0.755333333
toilet	62	0.758571429	0.742857143
tv	63	0.79035753	0.791440953
laptop	64	0.572222222	0.611111111
mouse	65	0.572192513	0.540106952
remote	66	0.62962963	0.759259259
keyboard	67	0.931972789	0.931972789
cell phone	68	0.715384615	0.684615385
microwave	69	1	1
oven	70	0.714285714	0.738095238
sink	72	0.773506494	0.770909091
refrigerator	73	0.710691824	0.757861635
book	74	0.49122807	0.539473684
clock	75	0.868050408	0.860637509
vase	76	0.704673857	0.769388803
teddy bear	78	0.616363636	0.627272727
Mean		0.669775013	**0.670430059**

Figure C.2.: AUC scores obtained from the Rescoring Model and YOLOv1

Appendix D

RELABELLING MODEL STATISTICAL RESULTS

D.1. Comparison Between Faster RCNN, Rescoring and Relabelling Models

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7), *Rescoring Model* and the *Relabelling Model* is presented in Figure D.1. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 4, Section 4.3, for more information.

D.2. Comparison Between Faster RCNN and Iterated Relabelling Model

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7) Iterated Relabelling Model is presented in Figure D.2. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 4, Section 4.3.2, for more information.

ClassName	Class_ID	Faster RCNN	Rescoring Model	Relabelling Model
person	1	0.84345527	0.846464998	0.845685171
bicycle	2	0.760735198	0.769843852	0.780063796
car	3	0.812464855	0.803651984	0.804743083
motorcycle	4	0.794412331	0.789788054	0.806264896
airplane	5	0.883006294	0.874861163	0.875091575
bus	6	0.89180145	0.883156721	0.89
train	7	0.930786095	0.931412465	0.926245211
truck	8	0.70042735	0.726353276	0.76535356
boat	9	0.699422808	0.677472015	0.68531746
traffic light	10	0.789794608	0.78690629	0.790837625
fire hydrant	11	0.940700809	0.894878706	0.866995074
stop sign	12	0.930127042	0.833030853	0.851393189
parking meter	13	0.965909091	0.818181818	0.915343915
bench	14	0.778590426	0.755718085	0.767656766
bird	15	0.801316939	0.8183162	0.842297704
cat	16	0.877673448	0.893192488	0.873536488
dog	17	0.80846395	0.75015674	0.751826087
horse	18	0.922232734	0.896499527	0.885352622
sheep	19	0.785630744	0.838596491	0.833967466
cow	20	0.823016661	0.845305087	0.840354767
elephant	21	0.740588235	0.749294118	0.753371429
bear	22	0.947916667	0.899305556	0.882758621
zebra	23	0.898245614	0.879622132	0.885897436
giraffe	24	0.857692308	0.791346154	0.788975155
backpack	25	0.614945511	0.688116243	0.716304348
umbrella	26	0.739066402	0.752257156	0.739321357
handbag	27	0.674710425	0.732625483	0.795093795
tie	28	0.845217391	0.850434783	0.86512474
suitcase	29	0.695374801	0.767676768	0.786412512
frisbee	30	0.901909722	0.861979167	0.849184783
skis	31	0.726060606	0.784848485	0.801412429
snowboard	32	0.742296919	0.826330532	0.874125874
sports ball	33	0.842447917	0.857638889	0.842032967
kite	34	0.738661504	0.768528761	0.782929782
baseball bat	35	0.823369565	0.795289855	0.814542484
baseball glove	36	0.831168831	0.849721707	0.856060606
skateboard	37	0.840243902	0.893170732	0.891666667
surfboard	38	0.784833716	0.810923165	0.811827957
tennis racket	39	0.858950617	0.814814815	0.825103448
bottle	40	0.748161061	0.751959856	0.74602329
wine glass	41	0.771267361	0.770833333	0.799874726
cup	42	0.773542882	0.753936884	0.749531381
fork	43	0.668478261	0.796195652	0.815455594
knife	44	0.695959596	0.594949495	0.569444444
spoon	45	0.654901961	0.611764706	0.6675
bowl	46	0.713890363	0.752634771	0.761153285
banana	47	0.685855263	0.659333882	0.686751428
apple	48	0.676910299	0.693521595	0.743873518
sandwich	49	0.767255892	0.741582492	0.741922121
orange	50	0.686057692	0.720673077	0.741869919
broccoli	51	0.639451399	0.666697349	0.685300391
carrot	52	0.593954248	0.609068627	0.632316958
hot dog	53	0.696969697	0.776859504	0.789473684
pizza	54	0.804299481	0.745552261	0.74309146
donut	55	0.737295826	0.744782214	0.734733701
cake	56	0.778571429	0.747402597	0.805190058
chair	57	0.751408724	0.762028334	0.759833313
couch	58	0.761538462	0.837948718	0.859953704
potted plant	59	0.713214245	0.735886866	0.744427448
bed	60	0.709486166	0.70685112	0.752571429
dining table	61	0.740068388	0.72406639	0.719761568
toilet	62	0.888790821	0.892321271	0.89500913
tv	63	0.860759494	0.845723477	0.83290653
laptop	64	0.854525627	0.816139586	0.856811594
mouse	65	0.864417989	0.922619048	0.925191816
remote	66	0.703703704	0.738425926	0.752777778
keyboard	67	0.759250694	0.79879741	0.804347826
cell phone	68	0.718074866	0.700962567	0.723013644
microwave	69	0.91954023	0.913793103	0.948051948
oven	70	0.763448276	0.8	0.835416667
sink	72	0.761671764	0.787665647	0.797568957
refrigerator	73	0.870634921	0.932539683	0.937037037
book	74	0.651536797	0.655779221	0.669411765
clock	75	0.923130314	0.905431199	0.912016088
vase	76	0.749644381	0.721479374	0.775206186
scissors	77	0.857142857	0.805194805	0.875
teddy bear	78	0.800490884	0.780154278	0.778971354
toothbrush	80	0.625	0.916666667	0.892857143
Mean		0.764728638	0.770574179	**0.782776522**

Figure D.1.: AUC scores: Faster RCNN, Rescoring and Relabelling Models

ClassName	Class_ID	Faster RCNN	Rescoring Model	IRM
person	1	0.84345527	0.846464998	0.999559762
bicycle	2	0.760735198	0.769843852	0.991704619
car	3	0.812464855	0.803651984	0.99899278
motorcycle	4	0.794412331	0.789788054	0.961494253
airplane	5	0.883006294	0.874861163	1
bus	6	0.89180145	0.883156721	0.813162298
train	7	0.930786095	0.931412465	1
truck	8	0.70042735	0.726353276	1
boat	9	0.699422808	0.677472015	1
traffic light	10	0.789794608	0.78690629	1
fire hydrant	11	0.940700809	0.894878706	1
stop sign	12	0.930127042	0.833030853	0.987621574
parking meter	13	0.965909091	0.818181818	1
bench	14	0.778590426	0.755718085	1
bird	15	0.801316939	0.8183162	1
cat	16	0.877673448	0.893192488	1
dog	17	0.80846395	0.75015674	0.711598746
horse	18	0.922232734	0.896499527	0.894628806
sheep	19	0.785630744	0.838596491	1
cow	20	0.823016661	0.845305087	0.983752621
elephant	21	0.740588235	0.749294118	0.962823529
bear	22	0.947916667	0.899305556	0.951388889
zebra	23	0.898245614	0.879622132	0.979487179
giraffe	24	0.857692308	0.791346154	1
backpack	25	0.614945511	0.688116243	1
umbrella	26	0.739066402	0.752257156	1
handbag	27	0.674710425	0.732625483	1
tie	28	0.845217391	0.850434783	1
suitcase	29	0.695374801	0.767676768	1
frisbee	30	0.901909722	0.861979167	0.952302632
skis	31	0.726060606	0.784848485	1
snowboard	32	0.742296919	0.826330532	1
sports ball	33	0.842447917	0.857638889	0.826436741
kite	34	0.738661504	0.768528761	1
baseball bat	35	0.823369565	0.795289855	1
baseball glove	36	0.831168831	0.849721707	1
skateboard	37	0.840243902	0.893170732	0.975609756
surfboard	38	0.784833716	0.810923165	1
tennis racket	39	0.858950617	0.814814815	1
bottle	40	0.748161061	0.751959856	1
wine glass	41	0.771267361	0.770833333	1
cup	42	0.773542882	0.753936884	1
fork	43	0.668478261	0.796195652	1
knife	44	0.695959596	0.594949495	0.978787879
spoon	45	0.654901961	0.611764706	0.57254902
bowl	46	0.713890363	0.752634771	1
banana	47	0.685855263	0.659333882	1
apple	48	0.676910299	0.693521595	1
sandwich	49	0.767255892	0.741582492	1
orange	50	0.686057692	0.720673077	1
broccoli	51	0.639451399	0.666697349	1
carrot	52	0.593954248	0.609068627	0.959113792
hot dog	53	0.696969697	0.776859504	1
pizza	54	0.804299481	0.745552261	1
donut	55	0.737295826	0.744782214	0.969882148
cake	56	0.778571429	0.747402597	1
chair	57	0.751408724	0.762028334	1
couch	58	0.761538462	0.837948718	0.880426504
potted plant	59	0.713214245	0.735886866	1
bed	60	0.709486166	0.70685112	0.877487562
dining table	61	0.740068388	0.72406639	0.998810737
toilet	62	0.888790821	0.892321271	1
tv	63	0.860759494	0.845723477	1
laptop	64	0.854525627	0.816139586	0.848200654
mouse	65	0.864417989	0.922619048	1
remote	66	0.703703704	0.738425926	1
keyboard	67	0.759250694	0.79879741	0.96692877
cell phone	68	0.718074866	0.700962567	1
microwave	69	0.91954023	0.913793103	1
oven	70	0.763448276	0.8	1
sink	72	0.761671764	0.787665647	0.981384139
refrigerator	73	0.870634921	0.932539683	1
book	74	0.651536797	0.655779221	1
clock	75	0.923130314	0.905431199	1
vase	76	0.749644381	0.721479374	0.994543905
scissors	77	0.857142857	0.805194805	1
teddy bear	78	0.800490884	0.780154278	1
toothbrush	80	0.625	0.916666667	1
Mean		0.764728638	0.770574179	**0.953146808**

Figure D.2.: AUC scores: Faster RCNN and Iterated Relabelling Model

Appendix E

STATISTICAL RESULTS ON PERTURBED IMAGES

E.1. Entire Image Perturbation: AUC For Faster RCNN Vs. Relabelling Model

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7) and the Relabelling Model is presented in Figure E.1. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 5, Section 5.3.1, for more information.

E.2. Perturbation On Regions: AUC For Faster RCNN Vs. Relabelling Model

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7) and the *Relabelling Model* is presented in Figure E.2. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 5, Section 5.3.2, for more information.

		FFF		UAP	
ClassName	Class_ID	Faster RCNN	Relabelling Model	Detector	Relabelling Model
person	**1**	0.623897068	0.690686375	0.628810568	0.696921316
bicycle	**2**	0.444763467	0.513290689	0.495824312	0.605090312
car	**3**	0.519584047	0.60710042	0.543813791	0.622670567
motorcycle	**4**	0.428612717	0.508448199	0.374458092	0.508003557
airplane	**5**	0.130136986	0.260273973	0.15851272	0.236790607
bus	**6**	0.550920245	0.564161554	0.471920717	0.449976404
train	**7**	0.42961165	0.472330097	0.410622501	0.506567676
truck	**8**	0.308302808	0.371225071	0.302387268	0.379435379
boat	**9**	0.240638911	0.334391534	0.228544776	0.363888889
traffic light	**10**	0.397883295	0.518743961	0.481518352	0.603815789
fire hydrant	**11**	0.619496855	0.544811321	0.534591195	0.382075472
stop sign	**12**	0.76242236	0.773291925	0.691558442	0.714285714
parking meter	**13**	0.318181818	0.393939394	0.454545455	0.568181818
bench	**14**	0.425088652	0.531053459	0.427051672	0.51577381
bird	**15**	0.323937426	0.441643324	0.408460569	0.513978668
cat	**16**	0.368696047	0.41552795	0.414590105	0.47625
dog	**17**	0.384848485	0.449888889	0.411363636	0.44787234
horse	**18**	0.481436885	0.549244985	0.475827815	0.551976346
sheep	**19**	0.357753358	0.455263158	0.387218045	0.478289474
cow	**20**	0.303459119	0.296855346	0.299467828	0.333333333
elephant	**21**	0.336862745	0.417429194	0.245882353	0.280462185
bear	**22**	0.240740741	0.277777778	0.15	0.316666667
zebra	**23**	0.568031189	0.55154265	0.610297483	0.636498856
giraffe	**24**	0.533182504	0.491386218	0.492063492	0.429647436
backpack	**25**	0.219512195	0.244962884	0.19764508	0.262825904
umbrella	**26**	0.552117448	0.657348618	0.524540251	0.620706515
handbag	**27**	0.384453782	0.395502646	0.468045113	0.513736264
tie	**28**	0.516790385	0.585365854	0.646599777	0.709130435
suitcase	**29**	0.136842105	0.227192982	0.164912281	0.209728868
frisbee	**30**	0.507211538	0.643880208	0.580729167	0.739583333
skis	**31**	0.255944056	0.381168831	0.44969697	0.532601881
snowboard	**32**	0.273809524	0.318181818	0.349206349	0.44973545
sports ball	**33**	0.629528986	0.735616438	0.631111111	0.719178082
kite	**34**	0.151548673	0.270543616	0.146017699	0.208304969
baseball bat	**35**	0.619565217	0.730818414	0.570393375	0.688043478
baseball glove	**36**	0.497732426	0.581356867	0.534985423	0.666002987
skateboard	**37**	0.432592593	0.491176471	0.46	0.537682927
surfboard	**38**	0.39266055	0.503410962	0.447502548	0.521046951
tennis racket	**39**	0.700132275	0.778897849	0.730833333	0.805813953
bottle	**40**	0.341145886	0.452154195	0.331712818	0.456983332
wine glass	**41**	0.337121212	0.37755102	0.322916667	0.352040816
cup	**42**	0.277487196	0.346138501	0.293311037	0.393710815
fork	**43**	0.391304348	0.523913043	0.287267081	0.417701863
knife	**44**	0.130769231	0.169354839	0.23125	0.247983871
spoon	**45**	0.033333333	0.244444444	0.133333333	0.266666667
bowl	**46**	0.337920489	0.394122383	0.387555869	0.428774929
banana	**47**	0.270394737	0.353546911	0.244987469	0.34962406
apple	**48**	0.318237454	0.329910141	0.249694002	0.261517615
sandwich	**49**	0.134848485	0.116666667	0.103896104	0.127705628
orange	**50**	0.308064516	0.36511456	0.4459375	0.490333135
broccoli	**51**	0.330952381	0.453222453	0.351020408	0.455348552
carrot	**52**	0.217361111	0.262792398	0.263513514	0.361486486
hot dog	**53**	0.146464646	0.166666667	0.181818182	0.303030303
pizza	**54**	0.673930921	0.784645629	0.631850244	0.75875817
donut	**55**	0.373310811	0.478174603	0.342633929	0.401785714
cake	**56**	0.266666667	0.384976526	0.267857143	0.357234315
chair	**57**	0.368161574	0.449981943	0.386557838	0.471748136
couch	**58**	0.551428571	0.718872999	0.423181818	0.575849515
potted plant	**59**	0.296791678	0.365317609	0.271567299	0.36419329
bed	**60**	0.407958086	0.483809524	0.445094509	0.522153846
dining table	**61**	0.461443983	0.528308824	0.472086005	0.519869146
toilet	**62**	0.190938511	0.346534653	0.161812298	0.308580858
tv	**63**	0.33875296	0.421940129	0.384489462	0.448757764
laptop	**64**	0.373644034	0.354135338	0.448299792	0.455424275
mouse	**65**	0.015873016	0.174603175	0.174603175	0.44047619
remote	**66**	0.181481481	0.186363636	0.330246914	0.43030303
keyboard	**67**	0.289565217	0.369326326	0.348861284	0.457489879
cell phone	**68**	0.273065015	0.324404762	0.420588235	0.528571429
microwave	**69**	0.103448276	0.214285714	0	0.142857143
oven	**70**	0.120689655	0.559322034	0.120689655	0.118644068
sink	**72**	0.052293578	0.057272727	0.079510703	0.093506494
refrigerator	**73**	0.028571429	0.107142857	0.035714286	0.114285714
book	**74**	0.261650794	0.35082909	0.346153846	0.429592636
clock	**75**	0.427728614	0.539434524	0.439259855	0.561688312
vase	**76**	0.397432606	0.471246537	0.357696127	0.448684211
scissors	**77**	0.454545455	0.454545455	0.363636364	0.181818182
teddy bear	**78**	0.518115942	0.603571429	0.55786445	0.64479638
toothbrush	**80**	0.4	0.363636364	0.2	0.181818182
Mean		0.347147813	**0.420300157**	0.360500236	**0.43342962**

Figure E.1.: AUC scores: Faster RCNN and Relabelling Model on Perturbed Images

		FFF		UAP	
ClassName	Class_ID	Faster RCNN	Relabelling Model	Faster RCNN	Relabelling Model
person	**1**	0.185739377	0.268687631	0.146530663	0.234666508
bicycle	**2**	0.10754717	0.197142857	0.100943396	0.164285714
car	**3**	0.243237076	0.294856198	0.209674064	0.266528636
motorcycle	**4**	0.116763006	0.144508671	0.120561519	0.127993394
airplane	**5**	0.205479452	0.198630137	0.054794521	0.054794521
bus	**6**	0.098159509	0.186503067	0.068711656	0.072392638
train	**7**	0.055016181	0.22815534	0.038834951	0.519417476
truck	**8**	0.079059829	0.11039886	0.098290598	0.124685772
boat	**9**	0.088432836	0.112777778	0.094979647	0.134744268
traffic light	**10**	0.335160819	0.429259259	0.300438596	0.357380952
fire hydrant	**11**	0.132075472	0.132075472	0.179245283	0.16509434
stop sign	**12**	0.275210084	0.380252101	0.231092437	0.359243697
parking meter	**13**	0	0	0	0.136363636
bench	**14**	0.04331307	0.180555556	0.057919622	0.120659722
bird	**15**	0.137261948	0.208257216	0.123966942	0.297478992
cat	**16**	0.068075117	0.222619048	0.039906103	0.198809524
dog	**17**	0.105454545	0.108333333	0.081818182	0.075
horse	**18**	0.080573951	0.06162465	0.137748344	0.092592593
sheep	**19**	0.13994709	0.2	0.177836567	0.241176471
cow	**20**	0.150943396	0.335849057	0.125786164	0.159591195
elephant	**21**	0.086554622	0.245098039	0.071176471	0.220294118
bear	**22**	0.125	0.5625	0.166666667	0.166666667
zebra	**23**	0.191666667	0.189912281	0.071052632	0.186842105
giraffe	**24**	0.106837607	0.08974359	0.121794872	0.08974359
backpack	**25**	0.192411924	0.25203252	0.037940379	0.040650407
umbrella	**26**	0.080014615	0.16242284	0.102484472	0.140852974
handbag	**27**	0.306122449	0.280612245	0.196428571	0.246031746
tie	**28**	0.260869565	0.35326087	0.157004831	0.257246377
suitcase	**29**	0.140350877	0.157894737	0.162280702	0.179824561
frisbee	**30**	0.3421875	0.284375	0.3203125	0.251953125
skis	**31**	0.031404959	0.087603306	0.092929293	0.121212121
snowboard	**32**	0.19047619	0.19047619	0	0
sports ball	**33**	0.465811966	0.569020021	0.332070707	0.367372354
kite	**34**	0.325104797	0.391814159	0.222600408	0.279782165
baseball bat	**35**	0.232919255	0.253623188	0.139130435	0.258152174
baseball glove	**36**	0.31292517	0.346938776	0.265306122	0.265306122
skateboard	**37**	0.24	0.193333333	0.089	0.194
surfboard	**38**	0.158465388	0.213511259	0.114678899	0.152140673
tennis racket	**39**	0.248076923	0.250595238	0.27037037	0.270833333
bottle	**40**	0.208986115	0.277893025	0.156956104	0.24819928
wine glass	**41**	0.270833333	0.369387755	0.247916667	0.255102041
cup	**42**	0.212832034	0.265323818	0.20609825	0.280908326
fork	**43**	0.144927536	0.222826087	0.152173913	0.201086957
knife	**44**	0.113333333	0.150537634	0.173333333	0.274193548
spoon	**45**	0.1	0.133333333	0.166666667	0.111111111
bowl	**46**	0.114678899	0.164653784	0.097333716	0.126003086
banana	**47**	0.073099415	0.090643275	0.088815789	0.152834008
apple	**48**	0.111295681	0.221254355	0.098837209	0.207317073
sandwich	**49**	0	0	0.045454545	0.045454545
orange	**50**	0.138888889	0.221883469	0.060294118	0.172764228
broccoli	**51**	0.158035714	0.259684685	0.129726891	0.199258082
carrot	**52**	0.209325397	0.257440476	0.12191358	0.165849673
hot dog	**53**	0.060606061	0.090909091	0.015151515	0.03030303
pizza	**54**	0.117982456	0.127233115	0.072368421	0.142390289
donut	**55**	0.144432773	0.199579832	0.044642857	0.047619048
cake	**56**	0.135714286	0.133802817	0.128571429	0.104225352
chair	**57**	0.140465496	0.210855221	0.113574054	0.18713615
couch	**58**	0	0.067961165	0	0.058252427
potted plant	**59**	0.167828107	0.205634987	0.141014858	0.192682927
bed	**60**	0.015151515	0.015384615	0	0
dining table	**61**	0.045643154	0.108854167	0.028526971	0.030357143
toilet	**62**	0.038834951	0.04950495	0.097087379	0.099009901
tv	**63**	0.092694905	0.169986264	0.091160221	0.08974359
laptop	**64**	0.053435115	0.067669173	0.051526718	0.054511278
mouse	**65**	0.250793651	0.331746032	0.224489796	0.280045351
remote	**66**	0.121399177	0.127272727	0.105820106	0.155844156
keyboard	**67**	0.061594203	0.076923077	0.085748792	0.079365079
cell phone	**68**	0.208088235	0.254464286	0.128342246	0.142857143
microwave	**69**	0.172413793	0.160714286	0.068965517	0.107142857
oven	**70**	0.051724138	0.050847458	0.103448276	0.118644068
sink	**72**	0.057667104	0.077777778	0.073394495	0.116287879
refrigerator	**73**	0	0	0.021428571	0.028571429
book	**74**	0.202848485	0.369204066	0.159189189	0.309369369
clock	**75**	0.405899705	0.45952381	0.357200322	0.478896104
vase	**76**	0.245614035	0.3	0.201169591	0.246115288
scissors	**77**	0	0	0	0.090909091
teddy bear	**78**	0	0	0.054347826	0.054945055
toothbrush	**80**	0.116666667	0.210227273	0.2	0.318181818
Mean		0.143104809	**0.194702396**	0.120437469	**0.169941156**

Figure E.2.: AUC scores: Faster RCNN and Relabelling Model on Perturbed Regions

Appendix F

STATISTICAL RESULTS FOR TEDM

F.1. Natural images: AUC Scores For Faster RCNN Vs. TEDM

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7) and the *TEDM* is presented in Figure F.1. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 6, Section 6.4.1, for more information.

F.2. Perturbed Images: AUC Scores For Faster RCNN Vs. TEDM

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7) and the *TEDM* is presented in Figure F.2. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 6, Section 6.4.2, for more information.

F.3. Perturbed Regions: AUC Scores For Faster RCNN Vs. TEDM

A comparison between all MSCOCO 2017 object classes AUC scores for Faster RCNN (threshold is 0.7) and the *TEDM* is presented in Figure F.3. Please note that class number 71 and 79 are not added, as there are no results obtained by the detector. The reader is referred to Chapter 6, Section 6.4.2, for more information.

ClassName	Class_ID	Faster RCNN	TEDM
person	1	0.84345527	0.951745156
bicycle	2	0.760735198	0.913467794
car	3	0.812464855	0.932878788
motorcycle	4	0.794412331	0.954597701
airplane	5	0.883006294	0.940288022
bus	6	0.89180145	0.958655642
train	7	0.930786095	0.985583995
truck	8	0.70042735	0.887464387
boat	9	0.699422808	0.886252228
traffic light	10	0.789794608	0.961756525
fire hydrant	11	0.940700809	0.992924528
stop sign	12	0.930127042	0.974358974
parking meter	13	0.965909091	0.988636364
bench	14	0.778590426	0.899869792
bird	15	0.801316939	0.932283465
cat	16	0.877673448	0.975160051
dog	17	0.80846395	0.938484848
horse	18	0.922232734	0.973277075
sheep	19	0.785630744	0.927311738
cow	20	0.823016661	0.932467265
elephant	21	0.740588235	0.967294118
bear	22	0.947916667	1
zebra	23	0.898245614	0.981646424
giraffe	24	0.857692308	0.989423077
backpack	25	0.614945511	0.825116762
umbrella	26	0.739066402	0.922803904
handbag	27	0.674710425	0.797297297
tie	28	0.845217391	0.977391304
suitcase	29	0.695374801	0.901116427
frisbee	30	0.901909722	0.935855263
skis	31	0.726060606	0.877272727
snowboard	32	0.742296919	0.882352941
sports ball	33	0.842447917	0.950086806
kite	34	0.738661504	0.886924342
baseball bat	35	0.823369565	0.932065217
baseball glove	36	0.831168831	0.970315399
skateboard	37	0.840243902	0.966585366
surfboard	38	0.784833716	0.958741259
tennis racket	39	0.858950617	0.986111111
bottle	40	0.748161061	0.928201239
wine glass	41	0.771267361	0.90875
cup	42	0.773542882	0.879099958
fork	43	0.668478261	0.86548913
knife	44	0.695959596	0.836458333
spoon	45	0.654901961	0.831372549
bowl	46	0.713890363	0.857090348
banana	47	0.685855263	0.830068482
apple	48	0.676910299	0.800249169
sandwich	49	0.767255892	0.908742004
orange	50	0.686057692	0.858173077
broccoli	51	0.639451399	0.859942207
carrot	52	0.593954248	0.778784219
hot dog	53	0.696969697	0.86631016
pizza	54	0.804299481	0.94627193
donut	55	0.737295826	0.934561483
cake	56	0.778571429	0.83961039
chair	57	0.751408724	0.882852591
couch	58	0.761538462	0.838791571
potted plant	59	0.713214245	0.911178315
bed	60	0.709486166	0.925373134
dining table	61	0.740068388	0.912415206
toilet	62	0.888790821	0.971848739
tv	63	0.860759494	0.950758103
laptop	64	0.854525627	0.940676118
mouse	65	0.864417989	0.953042328
remote	66	0.703703704	0.806233062
keyboard	67	0.759250694	0.904486586
cell phone	68	0.718074866	0.906512605
microwave	69	0.91954023	0.89047619
oven	70	0.763448276	0.912466844
sink	72	0.761671764	0.956743257
refrigerator	73	0.870634921	0.947574335
book	74	0.651536797	0.785548387
clock	75	0.923130314	0.959924321
vase	76	0.749644381	0.919367089
scissors	77	0.857142857	0.931818182
teddy bear	78	0.800490884	0.957997312
toothbrush	80	0.625	0.866666667
Mean		0.764728638	**0.892222371**

Figure F.1.: AUC scores: Faster RCNN vs. TEDM on Natural Images

		FFF		UAP	
ClassName	Class_ID	Faster RCNN	TEDM	Detector	TEDM
person	**1**	0.623897068	0.695762185	0.628810568	0.733428712
bicycle	**2**	0.444763467	0.589001986	0.495824312	0.693889045
car	**3**	0.519584047	0.657418631	0.543813791	0.710073145
motorcycle	**4**	0.428612717	0.524237881	0.374458092	0.550505051
airplane	**5**	0.130136986	0.369863014	0.15851272	0.406392694
bus	**6**	0.550920245	0.684982662	0.471920717	0.618755478
train	**7**	0.42961165	0.620514985	0.410622501	0.567730005
truck	**8**	0.308302808	0.46212791	0.302387268	0.423766198
boat	**9**	0.240638911	0.356732537	0.228544776	0.356382979
traffic light	**10**	0.397883295	0.52027027	0.481518352	0.640013495
fire hydrant	**11**	0.619496855	0.698113208	0.534591195	0.679245283
stop sign	**12**	0.76242236	1	0.691558442	0.785714286
parking meter	**13**	0.318181818	0.363636364	0.454545455	0.5
bench	**14**	0.425088652	0.640919811	0.427051672	0.542119565
bird	**15**	0.323937426	0.444753623	0.408460569	0.542060606
cat	**16**	0.368696047	0.599207746	0.414590105	0.587323944
dog	**17**	0.384848485	0.553506494	0.411363636	0.525837321
horse	**18**	0.481436885	0.580910024	0.475827815	0.584388186
sheep	**19**	0.357753358	0.464271769	0.387218045	0.488671536
cow	**20**	0.303459119	0.374475891	0.299467828	0.351022013
elephant	**21**	0.336862745	0.504044118	0.245882353	0.402573529
bear	**22**	0.240740741	0.358333333	0.15	0.201388889
zebra	**23**	0.568031189	0.676315789	0.610297483	0.657489879
giraffe	**24**	0.533182504	0.705128205	0.492063492	0.679487179
backpack	**25**	0.219512195	0.286875726	0.19764508	0.288617886
umbrella	**26**	0.552117448	0.727850932	0.524540251	0.70613354
handbag	**27**	0.384453782	0.5	0.468045113	0.623214286
tie	**28**	0.516790385	0.655639572	0.646599777	0.764426877
suitcase	**29**	0.136842105	0.219298246	0.164912281	0.213225371
frisbee	**30**	0.507211538	0.699776786	0.580729167	0.686383929
skis	**31**	0.255944056	0.391608392	0.44969697	0.544242424
snowboard	**32**	0.273809524	0.279761905	0.349206349	0.428571429
sports ball	**33**	0.629528986	0.775977366	0.631111111	0.759615385
kite	**34**	0.151548673	0.219298246	0.146017699	0.247807018
baseball bat	**35**	0.619565217	0.8	0.570393375	0.734404537
baseball glove	**36**	0.497732426	0.604823748	0.534985423	0.686621315
skateboard	**37**	0.432592593	0.537321429	0.46	0.588837209
surfboard	**38**	0.39266055	0.524646465	0.447502548	0.593657505
tennis racket	**39**	0.700132275	0.781884058	0.730833333	0.820208333
bottle	**40**	0.341145886	0.480387163	0.331712818	0.441276353
wine glass	**41**	0.337121212	0.383680556	0.322916667	0.39375
cup	**42**	0.277487196	0.375213808	0.293311037	0.40607914
fork	**43**	0.391304348	0.541062802	0.287267081	0.440993789
knife	**44**	0.130769231	0.221568627	0.23125	0.325490196
spoon	**45**	0.033333333	0.2	0.133333333	0.333333333
bowl	**46**	0.337920489	0.482657637	0.387555869	0.566479821
banana	**47**	0.270394737	0.437467295	0.244987469	0.330716902
apple	**48**	0.318237454	0.455602537	0.249694002	0.355813953
sandwich	**49**	0.134848485	0.222732491	0.103896104	0.28358209
orange	**50**	0.308064516	0.48030303	0.4459375	0.6221875
broccoli	**51**	0.330952381	0.596508012	0.351020408	0.618113689
carrot	**52**	0.217361111	0.409090909	0.263513514	0.521367521
hot dog	**53**	0.146464646	0.251633987	0.181818182	0.358823529
pizza	**54**	0.673930921	0.877678571	0.631850244	0.817837637
donut	**55**	0.373310811	0.488240418	0.342633929	0.493434874
cake	**56**	0.266666667	0.414285714	0.267857143	0.385714286
chair	**57**	0.368161574	0.482494464	0.386557838	0.523988975
couch	**58**	0.551428571	0.754555287	0.423181818	0.610036004
potted plant	**59**	0.296791678	0.454246215	0.271567299	0.440752688
bed	**60**	0.407958086	0.616796968	0.445094509	0.649391929
dining table	**61**	0.461443983	0.703012597	0.472086005	0.712738637
toilet	**62**	0.190938511	0.262585034	0.161812298	0.235714286
tv	**63**	0.33875296	0.486263736	0.384489462	0.511803012
laptop	**64**	0.373644034	0.480534351	0.448299792	0.59056211
mouse	**65**	0.015873016	0.238095238	0.174603175	0.253968254
remote	**66**	0.181481481	0.240740741	0.330246914	0.37037037
keyboard	**67**	0.289565217	0.420692432	0.348861284	0.528656126
cell phone	**68**	0.273065015	0.45831202	0.420588235	0.619047619
microwave	**69**	0.103448276	0.2	0	0
oven	**70**	0.120689655	0.120689655	0.120689655	0.120689655
sink	**72**	0.052293578	0.085454545	0.079510703	0.109090909
refrigerator	**73**	0.028571429	0.14084507	0.035714286	0.197183099
book	**74**	0.261650794	0.417007407	0.346153846	0.500798771
clock	**75**	0.427728614	0.60252193	0.439259855	0.541118421
vase	**76**	0.397432606	0.524597701	0.357696127	0.495357143
scissors	**77**	0.454545455	0.431818182	0.363636364	0.454545455
teddy bear	**78**	0.518115942	0.687123	0.55786445	0.695639188
toothbrush	**80**	0.4	0.4	0.2	0.197874672
Mean		0.347147813	**0.476839216**	0.360500236	**0.487133417**

Figure F.2.: AUC scores: Faster RCNN vs. TEDM Perturbed Images

		FFF		UAP	
ClassName	Class_ID	Faster RCNN	TEDM	Detector	TEDM
person	**1**	0.185739377	0.319822539	0.146530663	0.283851048
bicycle	**2**	0.10754717	0.254716981	0.100943396	0.20754717
car	**3**	0.243237076	0.336560409	0.209674064	0.395116381
motorcycle	**4**	0.116763006	0.173286991	0.120561519	0.208718463
airplane	**5**	0.205479452	0.476307498	0.054794521	0.04109589
bus	**6**	0.098159509	0.151533742	0.068711656	0.194478528
train	**7**	0.055016181	0.087378641	0.038834951	0.029126214
truck	**8**	0.079059829	0.170940171	0.098290598	0.153846154
boat	**9**	0.088432836	0.13178733	0.094979647	0.11697861
traffic light	**10**	0.335160819	0.361161525	0.300438596	0.419199085
fire hydrant	**11**	0.132075472	0.390566038	0.179245283	0
stop sign	**12**	0.275210084	0.407142857142857`	0.231092437	0.435714286
parking meter	**13**	0	0	0	0
bench	**14**	0.04331307	0.237239583	0.057919622	0.2625
bird	**15**	0.137261948	0.197	0.123966942	0.128
cat	**16**	0.068075117	0.105633803	0.039906103	0.084507042
dog	**17**	0.105454545	0.236363636	0.081818182	0.209090909
horse	**18**	0.080573951	0.234177215	0.137748344	0.158227848
sheep	**19**	0.13994709	0.253172589	0.177836567	0.231327049
cow	**20**	0.150943396	0.201257862	0.125786164	0.169811321
elephant	**21**	0.086554622	0.132352941	0.071176471	0.104411765
bear	**22**	0.125	0	0.166666667	0
zebra	**23**	0.191666667	0.331578947	0.071052632	0.284210526
giraffe	**24**	0.106837607	0.307692308	0.121794872	0.269230769
backpack	**25**	0.192411924	0.243902439	0.037940379	0.235772358
umbrella	**26**	0.080014615	0.236120401	0.102484472	0.302187416
handbag	**27**	0.306122449	0.178571429	0.196428571	0.107142857
tie	**28**	0.260869565	0.405797101	0.157004831	0.376811594
suitcase	**29**	0.140350877	0.187719298	0.162280702	0.187719298
frisbee	**30**	0.3421875	0.5625	0.3203125	0.572916667
skis	**31**	0.031404959	0.218181818	0.092929293	0.2
snowboard	**32**	0.19047619	0.142857143	0	0.095238095
sports ball	**33**	0.465811966	0.603703704	0.332070707	0.594017094
kite	**34**	0.325104797	0.096491228	0.222600408	0.170175439
baseball bat	**35**	0.232919255	0.369565217	0.139130435	0.239130435
baseball glove	**36**	0.31292517	0.459183673	0.265306122	0.469387755
skateboard	**37**	0.24	0.4	0.089	0.37
surfboard	**38**	0.158465388	0.288811189	0.114678899	0.289256198
tennis racket	**39**	0.248076923	0.508333333	0.27037037	0.325
bottle	**40**	0.208986115	0.286087369	0.156956104	0.270914271
wine glass	**41**	0.270833333	0.197916667	0.247916667	0.25625
cup	**42**	0.212832034	0.287369104	0.20609825	0.262047875
fork	**43**	0.144927536	0.217391304	0.152173913	0.284324697
knife	**44**	0.113333333	0.091666667	0.173333333	0.033333333
spoon	**45**	0.1	0.066666667	0.166666667	0.266666667
bowl	**46**	0.114678899	0.26767851	0.097333716	0.238664674
banana	**47**	0.073099415	0.250549451	0.088815789	0.2
apple	**48**	0.111295681	0.343023256	0.098837209	0.244186047
sandwich	**49**	0	0	0.045454545	0.114427861
orange	**50**	0.138888889	0.37109375	0.060294118	0.345833333
broccoli	**51**	0.158035714	0.285309735	0.129726891	0.25543041
carrot	**52**	0.209325397	0.220833333	0.12191358	0.177287582
hot dog	**53**	0.060606061	0.147058824	0.015151515	0.088235294
pizza	**54**	0.117982456	0.515899123	0.072368421	0.252380952
donut	**55**	0.144432773	0.302631579	0.044642857	0.273109244
cake	**56**	0.135714286	0	0.128571429	0
chair	**57**	0.140465496	0.224610067	0.113574054	0.196763477
couch	**58**	0	0.178217822	0	0.158415842
potted plant	**59**	0.167828107	0.225806452	0.141014858	0.166982922
bed	**60**	0.015151515	0.119402985	0	0
dining table	**61**	0.045643154	0.199074074	0.028526971	0.156378601
toilet	**62**	0.038834951	0	0.097087379	0
tv	**63**	0.092694905	0.282051282	0.091160221	0.208791209
laptop	**64**	0.053435115	0.20610687	0.051526718	0.175572519
mouse	**65**	0.250793651	0	0.224489796	0.3690919
remote	**66**	0.121399177	0.166666667	0.105820106	0
keyboard	**67**	0.061594203	0.310869565	0.085748792	0.242236025
cell phone	**68**	0.208088235	0.331372549	0.128342246	0.277647059
microwave	**69**	0.172413793	0	0.068965517	0.091432567
oven	**70**	0.051724138	0	0.103448276	0
sink	**72**	0.057667104	0.043181818	0.073394495	0.036363636
refrigerator	**73**	0	0	0.021428571	0
book	**74**	0.202848485	0.340995025	0.159189189	0.274333333
clock	**75**	0.405899705	0.472807018	0.357200322	0.312865497
vase	**76**	0.245614035	0.284723687	0.201169591	0.241492685
scissors	**77**	0	0	0	0
teddy bear	**78**	0	0.225806452	0.054347826	0
toothbrush	**80**	0.116666667	0.3	0.2	0.2
Mean		0.143104809	**0.230586187**	0.120437469	**0.199912869**

Figure F.3.: AUC scores: Faster RCNN vs. TEDM Perturbed Regions

Bibliography

Akhtar, N., Liu, J., and Mian, A. (2018). Defense against universal adversarial perturbations. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3389–3398.

Anderson, P., Fernando, B., Johnson, M., and Gould, S. (2016). SPICE: semantic propositional image caption evaluation. *CoRR*, abs/1607.08822.

Apruzzese, G., Colajanni, M., Ferretti, L., Guido, A., and Marchetti, M. (2018). On the effectiveness of machine and deep learning for cyber security. In *2018 10th International Conference on Cyber Conflict (CyCon)*, pages 371–390.

Awad, M. and Khanna, R. (2015). *Support Vector Machines for Classification*, pages 39–66. Apress, Berkeley, CA.

Bar, M. and Ullman, S. (1996). Spatial context in recognition. *Perception*, 25(3):343–352. PMID: 8804097.

Barnea, E. and Ben-Shahar, O. (2018). Contextual object detection with a few relevant neighbors. In *ACCV*.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bell, S., Zitnick, C. L., Bala, K., and Girshick, R. B. (2015). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *CoRR*, abs/1512.04143.

Biederman, I., Mezzanotte, R. J., and Rabinowitz, J. C. (1982). Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive Psychology*, 14(2):143 – 177.

Bozcan, I. and Kalkan, S. (2019). Cosmo: Contextualized scene modeling with boltzmann machines. *Robotics and Autonomous Systems*, 113:132 – 148.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1983). Classification and regression trees.

CambridgeDictionary (2020). Cambridge dictionary | english dictionary, translations & thesaurus. https://dictionary.cambridge.org/. (Accessed on 15/02/2020).

Cao, X., Wei, X., Han, Y., and Chen, X. (2015). An object-level high-order contextual descriptor based on semantic, spatial, and scale cues. *IEEE Transactions on Cybernetics*, 45(7):1327–1339.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. M., and Zagoruyko, S. (2020). End-to-end object detection with transformers. *ArXiv*, abs/2005.12872.

Carrara, F., Falchi, F., Caldelli, R., Amato, G., and Becarelli, R. (2019). Adversarial image detection in deep neural networks. *Multimedia Tools and Applications*, 78(3):2815–2835.

Celikkanat, H., Orhan, G., Pugeault, N., Guerin, F., Şahin, E., and Kalkan, S. (2016). Learning context on a humanoid robot using incremental latent dirichlet allocation. *IEEE Transactions on Cognitive and Developmental Systems*, 8(1):42–59.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *CoRR*, abs/1405.3531.

Chaubey, A., Agrawal, N., Barnwal, K., Guliani, K. K., and Mehta, P. (2020). Universal adversarial perturbations: A survey. *ArXiv*, abs/2005.08087.

Chen, G., Ding, Y., Xiao, J., and Han, T. X. (2013). Detection evolution with multi-order contextual co-occurrence. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1798–1805.

Chen, L. (2019). Understanding the efficacy, reliability and resiliency of computer vision techniques for malware detection and future research directions. *CoRR*, abs/1904.10504.

Chen, Q., Song, Z., Dong, J., Huang, Z., Hua, Y., and Yan, S. (2015). Contextualizing object detection and classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):13–27.

Chen, X. and Gupta, A. (2017). Spatial memory for context reasoning in object detection. *CoRR*, abs/1704.04224.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Choi, M. J., Lim, J. J., Torralba, A., and Willsky, A. S. (2010). Exploiting hierarchical context on a large database of object categories. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 129–136.

Choi, M. J., Torralba, A., and Willsky, A. S. (2012a). Context models and out-of-context objects. *Pattern Recognition Letters*, 33(7):853 – 862. Special Issue on Awards from ICPR 2010.

Choi, M. J., Torralba, A., and Willsky, A. S. (2012b). A tree-based context model for

object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):240–252.

Chu, W. and Cai, D. (2016). Deep feature based contextual model for object detection. *CoRR*, abs/1604.04048.

Cinbis, R. G. and Sclaroff, S. (2012). Contextual object detection using set-based classification. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision – ECCV 2012*, pages 43–57, Berlin, Heidelberg. Springer Berlin Heidelberg.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Dai, B., Zhang, Y., and Lin, D. (2017). Detecting visual relationships with deep relational networks. *CoRR*, abs/1704.03114.

Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Desai, C., Ramanan, D., and Fowlkes, C. C. (2011). Discriminative models for multi-class object layout. *International Journal of Computer Vision*, 95(1):1–12.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Divvala, S. K., Hoiem, D., Hays, J. H., Efros, A. A., and Hebert, M. (2009). An empirical study of context in object detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1271–1278.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531 vol. 2.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II.

Forsyth, D. A. and Ponce, J. (2002). *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference.

Galleguillos, C. and Belongie, S. (2010). Context based object categorization: A critical survey. *Comput. Vis. Image Underst.*, 114(6):712–722.

Galleguillos, C., Rabinovich, A., and Belongie, S. (2008). Object categorization using co-occurrence, location and appearance. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Gauen, K., Dailey, R., Laiman, J., Zi, Y., Asokan, N., Lu, Y., Thiruvathukal, G. K., Shyu, M., and Chen, S. (2017). Comparison of visual datasets for machine learning. In *2017 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 346–355.

Girshick, R. B. (2015). Fast R-CNN. *CoRR*, abs/1504.08083.

Girshick, R. B., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524.

Girshick, R. B., Felzenszwalb, P. F., and McAllester, D. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/ rbg/latent-release5/.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572.

Gould, S., Rodgers, J., Cohen, D., Elidan, G., and Koller, D. (2008). Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316.

Griffin, G., Holub, A., and Perona, P. (2006). Caltech256 image dataset.

Gundimeda, V., Murali, R. S., Joseph, R., and Naresh Babu, N. T. (2019). An automated computer vision system for extraction of retail food product metadata. In Bapi, R. S., Rao, K. S., and Prasad, M. V. N. K., editors, *First International Conference on Artificial Intelligence and Cognitive Computing*, pages 199–216, Singapore. Springer Singapore.

Gupta, S., Hariharan, B., and Malik, J. (2015). Exploring person context and local scene context for object detection. *CoRR*, abs/1511.08177.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. *CoRR*, abs/1703.06870.

He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729.

Hu, H., Gu, J., Zhang, Z., Dai, J., and Wei, Y. (2018). Relation networks for object detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3588–3597.

Inoue, N., Furuta, R., Yamasaki, T., and Aizawa, K. (2017). Object detection refinement using markov random field based pruning and learning based rescoring. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1652–1656.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA. ACM.

Johnson, J., Karpathy, A., and Li, F. (2015). Densecap: Fully convolutional localization networks for dense captioning. *CoRR*, abs/1511.07571.

Johnson, J., Krishna, R., Stark, M., Li, L., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. (2015). Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678.

Karpathy, A. and Li, F. (2014). Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306.

Krasser, M. (2020). Image captioning transformer. https://github.com/krasserm/fairseq-image-captioning#image-captioning-transformer. (Accessed on 07/15/2020).

Krause, J., Johnson, J., Krishna, R., and Fei-Fei, L. (2016). A hierarchical approach for generating descriptive image paragraphs. *CoRR*, abs/1611.06607.

Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L., Shamma, D. A., Bernstein, M. S., and Li, F. (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *CoRR*, abs/1602.07332.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

Kurakin, A., Goodfellow, I. J., and Bengio, S. (2016). Adversarial examples in the physical world. *CoRR*, abs/1607.02533.

Ladicky, L., Russell, C., Kohli, P., and Torr, P. H. S. (2010). Graph cut based inference with co-occurrence statistics. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, pages 239–253, Berlin, Heidelberg. Springer Berlin Heidelberg.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, J., Wei, Y., Liang, X., Dong, J., Xu, T., Feng, J., and Yan, S. (2016). Attentive contexts for object detection. *CoRR*, abs/1603.07415.

Li, X., Jiang, S., and Han, J. (2019). Learning object context for dense captioning. In *AAAI*, pages 8650–8657.

Li, Z., Peng, C., Yu, G., Zhang, X., Deng, Y., and Sun, J. (2017). Light-head R-CNN: in defense of two-stage object detector. *CoRR*, abs/1711.07264.

Li Fei-Fei, Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178.

Lin, T., Goyal, P., Girshick, R. B., He, K., and Dollár, P. (2017). Focal loss for dense object detection. *CoRR*, abs/1708.02002.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Liu, L., Ouyang, W., Wang, X., Fieguth, P. W., Chen, J., Liu, X., and Pietikäinen, M. (2018a). Deep learning for generic object detection: A survey. *CoRR*, abs/1809.02165.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C. (2015). SSD: single shot multibox detector. *CoRR*, abs/1512.02325.

Liu, Y., Wang, R., Shan, S., and Chen, X. (2018b). Structure inference net: Object detection using scene-level context and instance-level relationships. *CoRR*, abs/1807.00119.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2.

Lu, C., Krishna, R., Bernstein, M., and Fei-Fei, L. (2016a). Visual relationship detection with language priors. In *European Conference on Computer Vision*.

Lu, C., Krishna, R., Bernstein, M. S., and Li, F. (2016b). Visual relationship detection with language priors. *CoRR*, abs/1608.00187.

Lu, Z., Ip, H. H. S., and Peng, Y. (2011). Contextual kernel and spectral methods for learning the semantics of images. *IEEE Transactions on Image Processing*, 20(6):1739–1750.

Luo, J. and Oubong, G. (2009). A comparison of sift, pca-sift and surf. *International Journal of Image Processing*, 3.

Luo, W., Li, Y., Urtasun, R., and Zemel, R. S. (2017). Understanding the effective receptive field in deep convolutional neural networks. *CoRR*, abs/1701.04128.

Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-svms for object detection and beyond. In *2011 International Conference on Computer Vision*, pages 89–96.

MATLAB (2017). *Deep Learning Toolbox R2017a*. The MathWorks Inc., Natick, Massachusetts, United States.

Mensink, T., Gavves, E., and Snoek, C. G. M. (2014). Costa: Co-occurrence statistics for zero-shot classification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2441–2448.

Meur, Y. L., Vignolle, J.-M., and Chanussot, J. (2008). Practical use of receiver operating characteristic analysis to assess the performances of defect detection algorithms. *J. Electronic Imaging*, 17:031104.

Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., and Frossard, P. (2016). Universal adversarial perturbations. *CoRR*, abs/1610.08401.

Mopuri, K. R., Garg, U., and Babu, R. V. (2017). Fast feature fool: A data independent approach to universal adversarial perturbations. *CoRR*, abs/1707.05572.

Mottaghi, R., Chen, X., Liu, X., Cho, N.-G., Lee, S.-W., Fidler, S., Urtasun, R., and Yuille, A. (2014). The role of context for object detection and semantic segmentation in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898.

Mummadi, C. K., Brox, T., and Metzen, J. H. (2019). Defending against universal perturbations with shared adversarial training. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4927–4936.

Møller, M. F. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525 – 533.

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int. J. Comput. Vision*, 42(3):145–175.

Oñoro-Rubio, D., López-Sastre, R. J., Redondo-Cabrera, C., and Gil-Jiménez, P. (2018). The challenge of simultaneous object detection and pose estimation: A comparative study. *Image and Vision Computing*, 79:109 – 122.

Papernot, N., McDaniel, P. D., Sinha, A., and Wellman, M. P. (2016). Towards the science of security and privacy in machine learning. *CoRR*, abs/1611.03814.

Papernot, N., McDaniel, P. D., Wu, X., Jha, S., and Swami, A. (2015). Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508.

Park, D., Ramanan, D., and Fowlkes, C. (2010). Multiresolution models for object detection. In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, pages 241–254, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pinheiro, P. H. O. and Collobert, R. (2014). Weakly supervised semantic segmentation with convolutional networks. *CoRR*, abs/1411.6228.

Prabhu, V. U. and Birhane, A. (2020). Large image datasets: A pyrrhic win for computer vision?

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. J. (2007). Objects in context. *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8.

Radford, A. (2018). Improving language understanding by generative pre-training.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

Redmon, J. and Farhadi, A. (2016). Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *CoRR*, abs/1804.02767.

Ren, K., Zheng, T., Qin, Z., and Liu, X. (2020). Adversarial attacks and defenses in deep learning. *Engineering*, 6(3):346 – 360.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2016). Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A. C., and Li, F. (2014). Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173.

Sejnowski, T. J. (1986). Higher-order boltzmann machines. *AIP Conference Proceedings*, 151(1):398–403.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229.

Shafahi, A., Najibi, M., Xu, Z., Dickerson, J. P., Davis, L. S., and Goldstein, T. (2018). Universal adversarial training. *CoRR*, abs/1811.11304.

Shen, Y. and Zhu, W. (2013). Medical image processing using a machine vision-based approach.

Shetty, S. (2016). Application of convolutional neural network for image classification on pascal VOC challenge 2012 dataset. *CoRR*, abs/1607.03785.

Shrivastava, A. and Gupta, A. (2016). Contextual priming and feedback for faster R-CNN. In *ECCV (1)*, volume 9905 of *Lecture Notes in Computer Science*, pages 330–348. Springer.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Soh, M. (2016). Learning cnn-lstm architectures for image caption generation.

Sultana, F., Sufian, A., and Dutta, P. (2019). A review of object detection models based on convolutional neural network. *CoRR*, abs/1905.01614.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014a). Going deeper with convolutions. *CoRR*, abs/1409.4842.

Szegedy, C., Toshev, A., and Erhan, D. (2013). Deep neural networks for object detection. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014b). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tatbul, N., Lee, T. J., Zdonik, S., and Gottschlich, J. (2018). A new model for evaluating range-based anomaly detection algorithms. *CoRR*, abs/1803.03639.

Tian, H., Wang, T., Liu, Y., Qiao, X., and Li, Y. (2020). Computer vision technology in agricultural automation —a review. *Information Processing in Agriculture*, 7(1):1 – 19.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.

van de Sande, K. E. A., Uijlings, J. R. R., Gevers, T., and Smeulders, A. W. M. (2011).

Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, pages 1879–1886.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2014). Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I.

Wang, H., Wang, Q., Gao, M., Li, P., and Zuo, W. (2018). Multi-scale location-aware kernel representation for object detection. *CoRR*, abs/1804.00428.

Wang, L., Guo, S., Huang, W., Xiong, Y., and Qiao, Y. (2016). Knowledge guided disambiguation for large-scale scene classification with multi-resolution cnns. *CoRR*, abs/1610.01119.

Wang, X., Shrivastava, A., and Gupta, A. (2017). A-fast-rcnn: Hard positive generation via adversary for object detection. *CoRR*, abs/1704.03414.

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. (2016). Sun database: Exploring a large collection of scene categories. *International Journal of Computer Vision*, 119(1):3–22.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

Zeng, X., Ouyang, W., Yang, B., Yan, J., and Wang, X. (2016). Gated bi-directional cnn for object detection. In *ECCV*.

Zhao, Z., Ma, H., and You, S. (2016). Single image action recognition using semantic body part actions. *CoRR*, abs/1612.04520.

Zhao, Z., Zheng, P., Xu, S., and Wu, X. (2018). Object detection with deep learning: A review. *CoRR*, abs/1807.05511.

Zhiqiang, W. and Jun, L. (2017). A review of object detection based on convolutional neural network. In *2017 36th Chinese Control Conference (CCC)*, pages 11104–11109.

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey. *CoRR*, abs/1905.05055.